SIEMENS

SIMOTION

SIMOTION IT Ethernet basierende HMI- und Diagnosefunktion

Diagnosehandbuch

11/2010

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠ GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠ VORSICHT
mit Warndreieck bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

VORSICHT
ohne Warndreieck bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠ WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Vorwort

SIMOTION Dokumentation

Einen Überblick zur SIMOTION Dokumentation erhalten Sie in einem separaten Literaturverzeichnis.

Diese Dokumentation ist als elektronische Dokumentation im Lieferumfang von SIMOTION SCOUT enthalten und besteht aus 10 Dokumentationspaketen.

Zur SIMOTION Produktstufe V4.2 stehen folgende Dokumentationspakete zur Verfügung:

- SIMOTION Engineering System Handhabung
- SIMOTION System- und Funktionsbeschreibungen
- SIMOTION Service und Diagnose
- SIMOTION IT
- SIMOTION Programmieren
- SIMOTION Programmieren - Referenzen
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Ergänzende Dokumentation

Hotline und Internetadressen

Weiterführende Informationen

Unter folgendem Link finden Sie Informationen zu den Themen:

- Dokumentation bestellen / Druckschriftenübersicht
- Weiterführende Links für den Download von Dokumenten
- Dokumentation online nutzen (Handbücher/Informationen finden und durchsuchen)

http://www.siemens.com/motioncontrol/docu

Bei Fragen zur technischen Dokumentation (z. B. Anregungen, Korrekturen) senden Sie bitte eine E-Mail an folgende Adresse:
docu.motioncontrol@siemens.com

My Documentation Manager

Unter folgendem Link finden Sie Informationen, wie Sie Dokumentation auf Basis der Siemens Inhalte individuell zusammenstellen und für die eigene Maschinendokumentation anpassen:

http://www.siemens.com/mdm

Training

Unter folgendem Link finden Sie Informationen zu SITRAIN - dem Training von Siemens für Produkte, Systeme und Lösungen der Automatisierungstechnik:

http://www.siemens.com/sitrain

FAQs

Frequently Asked Questions finden Sie in den Service&Support-Seiten unter **Produkt Support**:

http://support.automation.siemens.com

Technical Support

Landesspezifische Telefonnummern für technische Beratung finden Sie im Internet unter **Kontakt**:

http://www.siemens.com/automation/service&support

Inhaltsverzeichnis

Einleitung 1

1.1 Überblick

Funktionspakete

"SIMOTION IT Ethernet basierende HMI- und Diagnose-Funktionen" enthält folgende Funktionspakete (IT=Information Technologie):

- SIMOTION IT DIAG
  Dieses Funktionspaket ermöglicht eine direkte Diagnose der SIMOTION Geräte. Der Zugriff erfolgt mit einem Standardbrowser (z. B. Firefox) über die IP-Adresse des SIMOTION Geräts. Für den Zugriff können Sie die Diagnose-Standardseiten nutzen oder eigene HTML-Seiten erstellen.
- SIMOTION IT OPC XML-DA Server

  Dieses Funktionspaket umfasst einen Webservice, der es ermöglicht Applikationen über Internettechnik mit einer Steuerung zu verbinden und auf Daten und Betriebszustände im SIMOTION Gerät zuzugreifen. Zur Befehlsübermittlung dient das Kommunikationsprotokoll SOAP (Simple Object Access Protocol).
- Trace via SOAP (TVS)

  Dieses Funktionspaket umfasst einen Webservice, der es ermöglicht Variablen aus dem Haushalt des SIMOTION Variablen Providers zu tracen (aufzuzeichnen).

Funktionspakete in anderen Dokumenten

- SIMOTION VM

  Dieses Funktionspaket stellt ein Laufzeitsystem zur Ausführung von Java-Anwendungen auf dem SIMOTION Gerät zur Verfügung. Bei der Jamaica Virtual Machine (JamaicaVM), die dafür zum Einsatz kommt, handelt es sich um eine Implementierung der "Java Virtual Machine Specification".
  Die SIMOTION VM wird in einem eigenen Programmierhandbuch "SIMOTION – IT Virtual Machine" beschrieben.

1.2 Prinzipielle Darstellung der Funktionspakete im SIMOTION Gerät

Darstellung der Funktionspakete

Im folgenden Bild sind die Funktionspakete im SIMOTION Gerät prinzipiell dargestellt.

Die Daten des SIMOTION Geräts sind über "Variablen Provider" erreichbar.

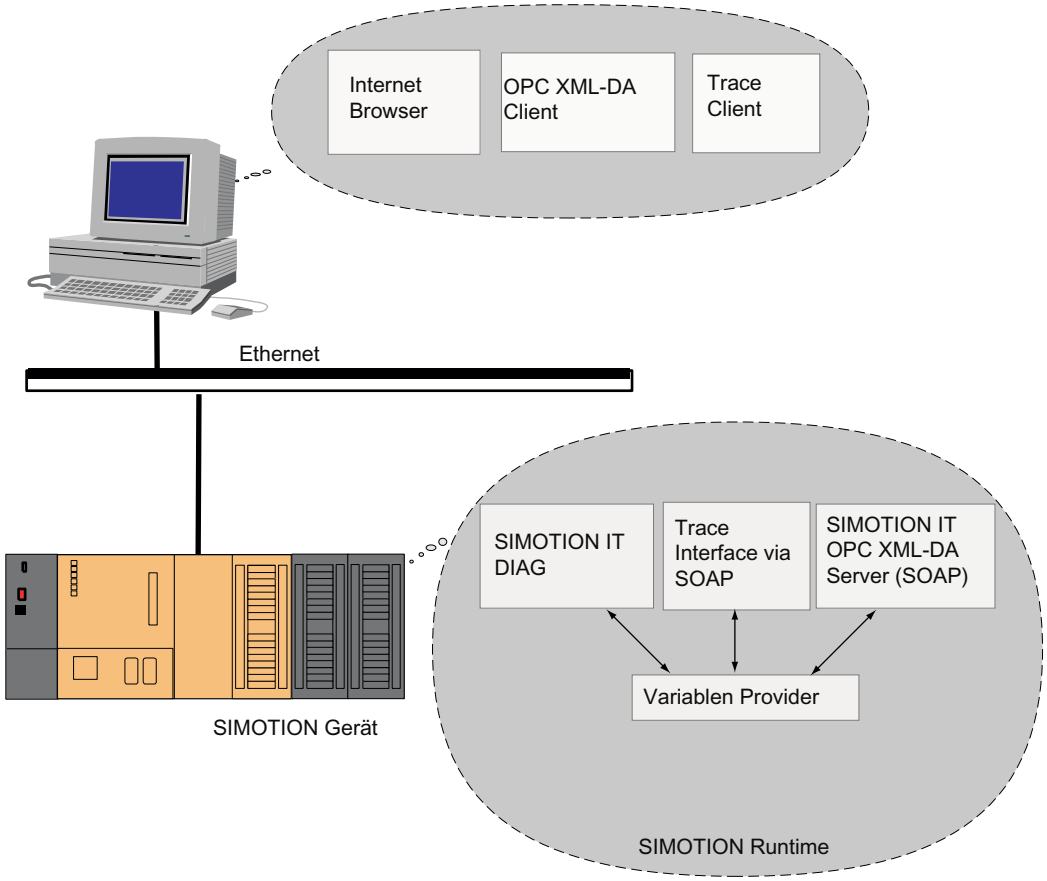


Bild 1-1 Funktionspakete

1.3 Lieferform

Lieferform

In der Firmware der Steuerung sind die "SIMOTION IT Ethernet basierende HMI- und Diagnose-Funktionen" bereits enthalten.

Hinweis

Die Funktionalitäten müssen im SIMOTION SCOUT Projekt unter der Hardwarekonfiguration der Steuerung aktiviert sein. Über die Objekteigenschaften der Steuerung in der Hardwarekonfiguration können Sie über den Reiter "Ethernet erweitert" die Funktion "OPC XML/Diagnoseseiten" aktivieren.

Diese Einstellungen sind ab V4.1.2 so voreingestellt.

Dokumentation, Tools, Beispiele und Konfigurationsdateien

Die Dokumentation, Tools, Beispiele, Konfigurationsdateien und andere Zusätze finden Sie auf der DVD "Documentation, Utilities & Applications".

Laufzeitlizenzen vor der Version 4.2

Bei den älteren Versionen erfordert der Zugriff z. B. auf die Watch Seite eine OPC XML-DA Einfach-Lizenz.

Beim Öffnen einer entsprechenden Seite erscheint dieser Hinweis:

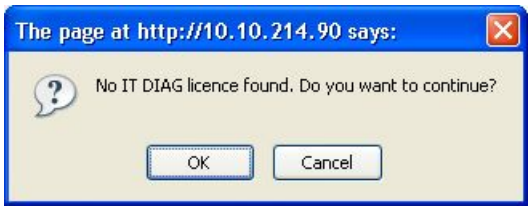


Bild 1-2 Warnung wegen fehlender Lizenz

Beim Anklicken des **OK** Buttons wird die gewünschte Seite geöffnet. Ein Weiterarbeiten ist somit auch ohne vorhandene Lizenz möglich. Es wird jedoch ein Eintrag im Diagnosepuffer gemacht und die Fehler-LED der Steuerung beginnt zu blinken.

1.4 Anwendungsmöglichkeiten

1.4.1 Standard Informationen

Anwendung der Diagnoseseiten

Die von IT DIAG bereitgestellten Webseiten liefern Informationen zu einem SIMOTION Gerät. Der Zugriff auf die Informationen erfolgt über den Webbrowser und das Ethernet.

Bild 1-3 Diagnose Standardseite 'Home' im Internet Explorer

Das SIMOTION Gerät wird dazu an das lokale Ethernet angeschlossen. Der Zugriff auf die Diagnoseseiten kann dann von jedem Rechner im Netz über die entsprechende IP-Adresse des Geräts erfolgen.

Es werden auch HTTPS-Verbindungen unterstützt, was eine Nutzung von IT DIAG über offene Netze (Internet) ermöglicht. Siehe Secure Socket Layer (Seite 265)

Die Nutzung der IT DIAG Standardseiten bedarf keiner speziellen Installation. Das Gerät ist schon entsprechend eingerichtet.

1.4.2 Benutzerdefinierte Informationen

Informationen in selbst erstellten Seiten anzeigen

Neben der Anzeige der Standardseiten bietet IT DIAG die Möglichkeit, eigene Webseiten zu erstellen.

Mit der Hilfe einer JavaScript-Bibliothek können Gerätedaten in einer Webseite abgefragt und dargestellt werden.

Eine weitere Möglichkeit besteht in der Benutzung der MiniWeb Server Language (MWSL). Eine an ECMA-Skript angelehnte Sprache, die serverseitig ausgeführt wird.

Über die "Variablen Provider" können folgende Informationen in einer Webseite gelesen und geschrieben werden:

- Systemvariablen des SIMOTION Geräts
- Systemvariablen und Konfigurationsdaten der Technologieobjekte
- Globale Unit-Variablen
- Antriebsparameter
- IO-Variablen
- Geräteglobale Variable

Durch die selbst erstellten Seiten ergeben sich vielfältige Möglichkeiten, Geräteinformationen darzustellen.

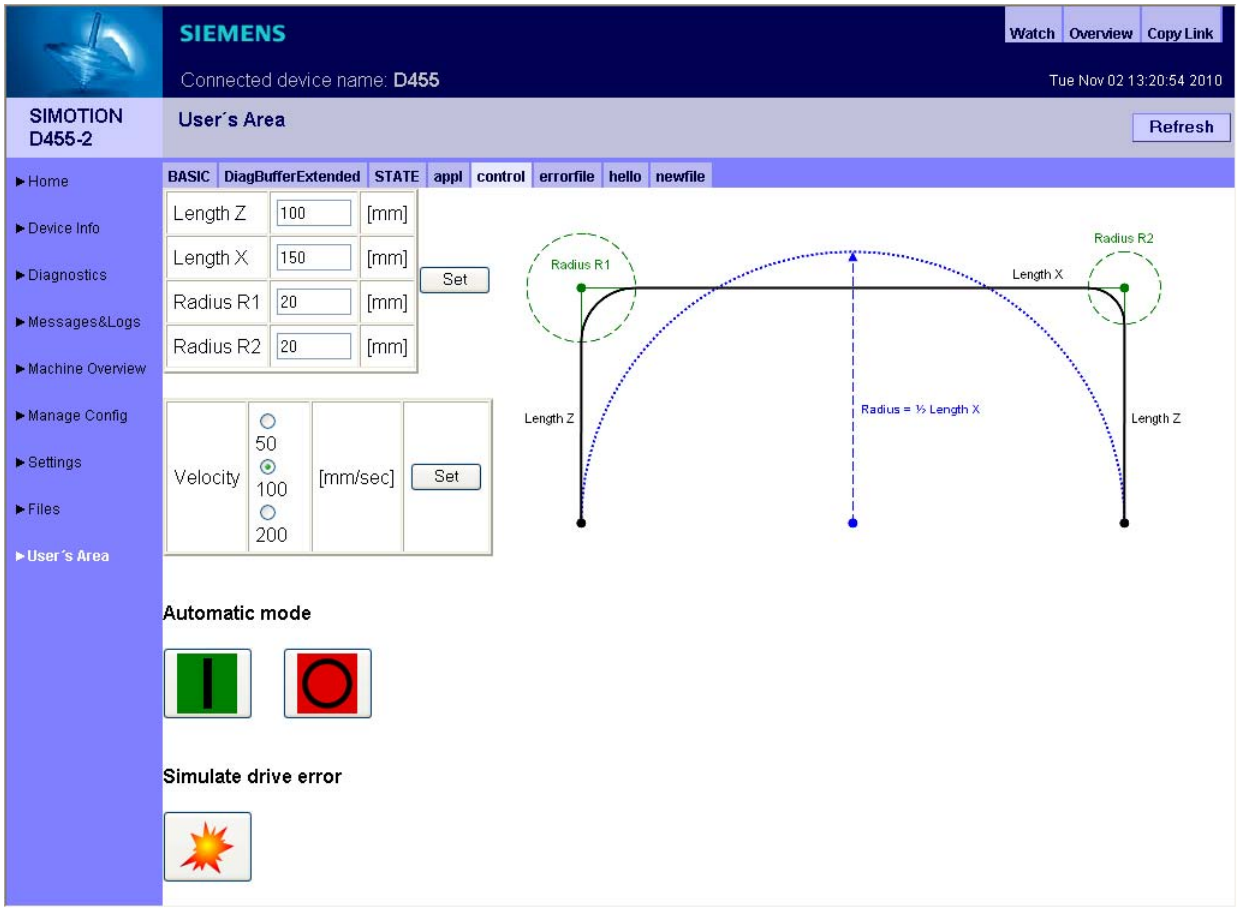


Bild 1-4 Beispiel einer benutzerdefinierten SIMOTION IT DIAG Webseite

MWSL

Die MWSL wird serverseitig ausgeführt. Sie sollten die MWSL verwenden, wenn die erstellten Seiten auf Geräten zur Anzeige kommen, die JavaScript nicht unterstützen. Außerdem können Variablenfunktionen schneller und direkter (systemnaher) ausgeführt werden, als mit Hilfe von JavaScript.

Ein Nachteil beim Einsatz der MWSL sind einerseits die fehlende Dynamik, da die Seiten immer statisch erzeugt werden. Andererseits belastet die Auswertung den Server und kann somit bei ausgelasteten Steuerungen mitunter lange dauern und andere Web-Prozesse bzw. –anfragen aufhalten.

JavaScript

SIMOTION IT unterstützt Sie bei der Erstellung dynamischer und flexibler Webseiten mit einer umfangreichen JavaScript-Bibliothek. Im Gegensatz zur MWSL wird diese im Browser abgearbeitet. Der Einsatz von JavaScript, entlastet die Steuerung und bringt erheblich mehr Möglichkeiten als die MWSL. Zum Anzeigen wird jedoch ein moderner Browser mit entsprechender JavaScript-Unterstützung benötigt, was nicht auf allen Automatisierungsumgebungen gewährleistet werden kann.

Inbetriebnehmen

2

2.1 Hard- und Softwarevoraussetzungen

Hardwarevoraussetzungen

- SIMOTION Gerät
- PC oder Notebook mit Ethernet-Anschluss.

Softwarevoraussetzungen

- Browser: Firefox ab Version 3.

2.2 SIMOTION Geräte-Schnittstelle konfigurieren

Konfiguration der Ethernet-Schnittstelle

Der Zugriff auf SIMOTION IT DIAG funktioniert über jede vorhandene Ethernet Schnittstelle der SIMOTION inklusive des PROFINET IO IRT.

Um über einen Browser mit den Diagnose-Standardseiten zu einem SIMOTION Gerät eine Verbindung herzustellen, müssen folgende Schritte zur Konfiguration der Ethernet-Schnittstelle ausgeführt werden:

Tabelle 2- 1 Schnittstelle konfigurieren

Schritt	Vorgehen
1	Die Funktionalität muss im SIMOTION SCOUT Projekt unter der Hardwarekonfiguration der CPU aktiviert sein. Über die Objekteigenschaften der CPU in der Hardwarekonfiguration können Sie über den Reiter "Ethernet erweitert" die Funktion "OPC XML/Diagnoseseiten" aktivieren. Ab V4.1.2 sind dies die Standardeinstellungen.
2	Eventuell müssen USER NAME und PASSWORT in der Konfigurationsdatei WebCfg.xml geändert werden. Die Voreinstellung sind USER NAME "simotion" und PASSWORT "simotion". Diese Einstellungen sollten umgehend geändert werden.
3	Zur Anzeige der Diagnose-Standardseiten im Browser muss die IP-Adresse des SIMOTION Geräts, wie z. B. http://169.254.11.22, eingegeben werden. In den Handbüchern zu den jeweiligen Steuerungen sind die voreingestellten IP-Adressen vermerkt. Diese Werkseinstellung kann in der HW Konfig geändert und anschließend in das SIMOTION Gerät geladen werden.

2.3 Loginverwaltung

Aufbau der Loginverwaltung

Das Loginsystem ist folgendermaßen aufgebaut:

- Es gibt Benutzer (User).
- Jeder User hat ein Passwort. Dieses kann wahlweise als Klartext oder als MD5 Hash vorliegen.
- Es gibt Sicherheitsbereiche (SecureGroups bzw. Realms).
- Jeder Sicherheitsbereich hat eine Gruppe von Benutzern, die diesen "betreten" dürfen.
- Ein Benutzer kann in unterschiedlichen Sicherheitsbereichen Zutritt haben.

Die User Database ist Bestandteil der Datei WebCfg.xml und kann nur durch erneutes Laden der WebCfg.xml verändert werden.

Hinweis

Fehlerhafte Änderungen in der Datei WebCfg.xml können zum Absturz des Geräts führen.

Hinweis

Bei der Auslieferung des Geräts sind der Benutzer "simotion" und das Passwort "simotion" für die Anmeldung voreingestellt.

Bitte ändern Sie umgehend diese Zugangsdaten.

Beispielkonfiguration

```
<SERVERPAGES>
  <BASE LOCALLINK="/">
    <FILES LOCALLINK="FILES/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE"
           READ="Anyone" WRITE="Anyone" MODIFY="Anyone"/>
    <SETTINGS.MCS REALM="Administrator,Servicegroup"
                  READ="Administrator,Servicegroup"
                  WRITE="Administrator,Servicegroup"
                  MODIFY="Administrator,Servicegroup"/>

    <UserDataBase>
      <FILE NAME="UserDataBase.xml">
        <![CDATA[
          <?xml version="1.0" encoding="UTF-8"?>
          <UserDataBase>
            <USER NAME="anonymous" PASSWORD="anonymous">
              <DESCRIPTION>Anonymous</DESCRIPTION>
              <GROUP NAME="Anyone"/>
              <GROUP NAME="OPC_XML"/>
            </USER>
            <USER NAME="internal" PASSWORD="internal">
              <DESCRIPTION>Internal user</DESCRIPTION>
              <GROUP NAME="Anyone"/>
            </USER>
            <USER NAME="simotion" PASSWORD="simotion">
              <DESCRIPTION>Default User</DESCRIPTION>
              <GROUP NAME="Administrator"/>
              <GROUP NAME="FTPUser"/>
              <GROUP NAME="Anyone"/>
              <GROUP NAME="OPC_XML"/>
            </USER>
            <USER NAME="user1" PASSWORD="user1">
              <DESCRIPTION>Default User</DESCRIPTION>
              <GROUP NAME="Administrator"/>
              <GROUP NAME="FTPUser"/>
              <GROUP NAME="Anyone"/>
              <GROUP NAME="OPC_XML"/>
              <GROUP NAME="Servicegroup"/>
    </UserDataBase>
    …
  </BASE LOCALLINK>
<SERVERPAGES>
```

Bei dem USER "simotion" ist das Passwort im Klartext angegeben ("simotion"). Der USER "simotion" darf nur den Sicherheitsbereich "Administrator" betreten.

Für das Update der WebCfg.xml ist die Gruppe "Administrator" fest einprogrammiert. Welche User zu dieser Gruppe gehören, wird in der USERDATABASE festgelegt.

Der User1 wurde eingefügt. Er gehört der neuen Gruppe "Servicegroup" an und hat Zugriff auf die Seite "Settings.mcs".

Sicherheitsbereiche

Weitere Sicherheitsbereiche können in der WebCfgFrame.xml den jeweiligen Anforderungen angepasst werden.

- Der Zugriff auf die Settings:

```
<settings.mcs SECUREGROUP="Administrator">
```

- Der Zugriff auf die Dateiablage "Files":

```
<DEFAPP SECUREGROUP="Administrator">
```

- Das Schreiben von Variablen in den HTML-Diagnoseseiten:

```
<VarApp SECUREGROUP="Administrator">
```

- Das Update von Projekt und Firmware:

```
<FWUpdtApp SECUREGROUP="Administrator">
```

- Den Zugriff auf den OPC XML-DA Service schützen:

```
<URL BASE="OpcXml" SECUREGROUP="Administrator"/>
```

MD5 Hash

MD5 (Message-Digest Algorithm 5) ist ein kryptografisches Hashverfahren, bei dem eine zu schützende Zeichenkette, wie hier das Passwort, nicht im Klartext in der Konfiguration gespeichert wird.

Die Speicherung des Passworts im Klartext hätte den Nachteil, dass ein Angreifer es lesen und benutzen könnte, um sich unerlaubten Zugriff zum System zu verschaffen. Stattdessen wird das Passwort als so genannter Hash hinterlegt. Der Hash ist eine nahezu eineindeutige Art Fingerabdruck des Passworts.

Um eine Authentifizierung vorzunehmen, sendet der Client (Webbrowser) das Passwort zum Server, der dann den Hash und das MD5 erzeugt. Dieser Hash kann mit dem in der Konfiguration hinterlegten verglichen werden und es kann entsprechend reagiert werden. Dieses Verfahren gilt als eines der sichersten seiner Art. Weitere Informationen finden Sie im Internet, z. B. unter http://de.wikipedia.org/wiki/Message-Digest_Algorithm_5.

Im Internet existieren viele Seiten, die erklären, wie ein MD5-Hash Ihrer Passworte erzeugt werden kann. Alternativ bieten alle gängigen Programmierframeworks, die sich mit Internettechnologie beschäftigen (z. B. Microsoft .NET oder Java) entsprechende Implementierungen, sodass Sie selbst einfach ein Programm erstellen können, das die Umwandlung für Sie vornimmt.

Einbinden des MD5 in die WebCfg.xml

Statt des Attributs PASSWORD="..." wird das Attribut MD5="..." verwendet. Der Attributwert ist der 32 Zeichen lange MD5-Hash, wobei Groß- und Kleinschreibung nicht unterschieden wird.

Beispiel: <USER NAME="simotion" MD5="5fc8f76e94ad3ab985ad8b4f192dc9ef">

Ein Angreifer kann das im Beispiel verschlüsselte Passwort "simotion" nicht mehr lesen. Ein Benutzer, der das Passwort kennt, kann sich damit im Browser authentifizieren.

Hinweis

MD5 und Klartext-Passwörter können gemischt verwendet werden; werden bei einem Benutzer beide Verfahren konfiguriert, wird immer das MD5-Passwort verwendet.

Hinweis

Die MD5-Passwörter können mit dem Benutzereditor in IT DIAG nicht bearbeitet werden.

Siehe auch

IT DIAG System (Seite 81)

2.4 Spracheinstellung der AlarmS- und der benutzerdefinierten Diagnosepuffer-Meldungen

Die Einstellung der Sprache der AlarmS- und der benutzerdefinierten Diagnosepuffermeldungen ist in jeder Sprache des SIMOTION SCOUT möglich.

Sprachauswahl

IT DIAG benutzt 4 Regeln zur Sprachauswahl. Die Regel, die zuerst zutrifft, kommt zur Anwendung:

1. Konfigurationskonstante ForceUserMsgLanguageID

Es existiert ein Eintrag für die gewünschte Sprache in den Konfigurationskonstanten. Die Variable `ForceUserMsgLanguageID` wird dazu auf den entsprechenden Ländercode (Dezimalwert) gesetzt. Die gewählte Sprache muss existieren. Trifft dies nicht zu, wird die THX-Darstellung verwendet.
Weitere Informationen zu den Konfigurationskonstanten (Seite 170) und Ländercodes (Seite 308).

2. SIMOTION SCOUT-Export

Durch einen SCOUT-Export benutzerdefinierter AlarmS - und Diagnosepuffermeldungen und den Upload (Seite 84) dieser Daten, wird die im SCOUT eingestellte Sprache in IT DIAG gesetzt.

3. Sprache System-Diagnosepuffertexte

Es wird versucht, die passende Sprache zu den installierten System-Diagnosepuffertexten, zu finden.

4. Sonstige Spracheinstellungen

Wird keine passende Sprache bei den System-Diagnosepuffertexten gefunden, wird der System-Default gewählt.

Im Syslog wird vermerkt, welche Sprache ausgewählt wurde.

Bedienen (Software) 3

3.1 IT DIAG Überblick und allgemeine Funktionen

3.1.1 Überblick

Das SIMOTION Gerät verwaltet vorgefertigte Diagnose-Standardseiten. Diese Seiten können über einen handelsüblichen Browser via Ethernet angezeigt werden. Darüber hinaus können Sie eigene HTML Seiten erstellen und Service- und Diagnoseinformationen einbinden.

Ziel und Nutzen

Das Ziel und der Nutzen der HTML-Diagnoseseiten ist folgender:

- Zur direkten Diagnose des SIMOTION Geräts stehen dem Anwender vorgefertigte Diagnoseseiten zur Verfügung.
- Ohne herstellerspezifische Programme kann auf Service- und Diagnoseinformationen der Geräte zur Diagnose bzw. Produktionsüberwachung zugegriffen werden.
- Anwenderdefinierte HTML Seiten können eingebunden werden.

3.2 Standardseiten

3.2.1 Home

Daten des SIMOTION Geräts

Auf der Startseite werden folgende aktuellen Daten des SIMOTION Geräts angezeigt:

Order Number	Bestellnummer (MLFB) des Geräts
Revision Number	Hardwareversion
Licence Serial Number	An diese Seriennummer wird der Licence Key gebunden.
User Version Firmware	SIMOTION Kernel Anwender Version
Operating State	Betriebszustand des SIMOTION Geräts (RUN, STOP, STOPU)
Systemtime	Aktuelle Uhrzeit des SIMOTION Geräts

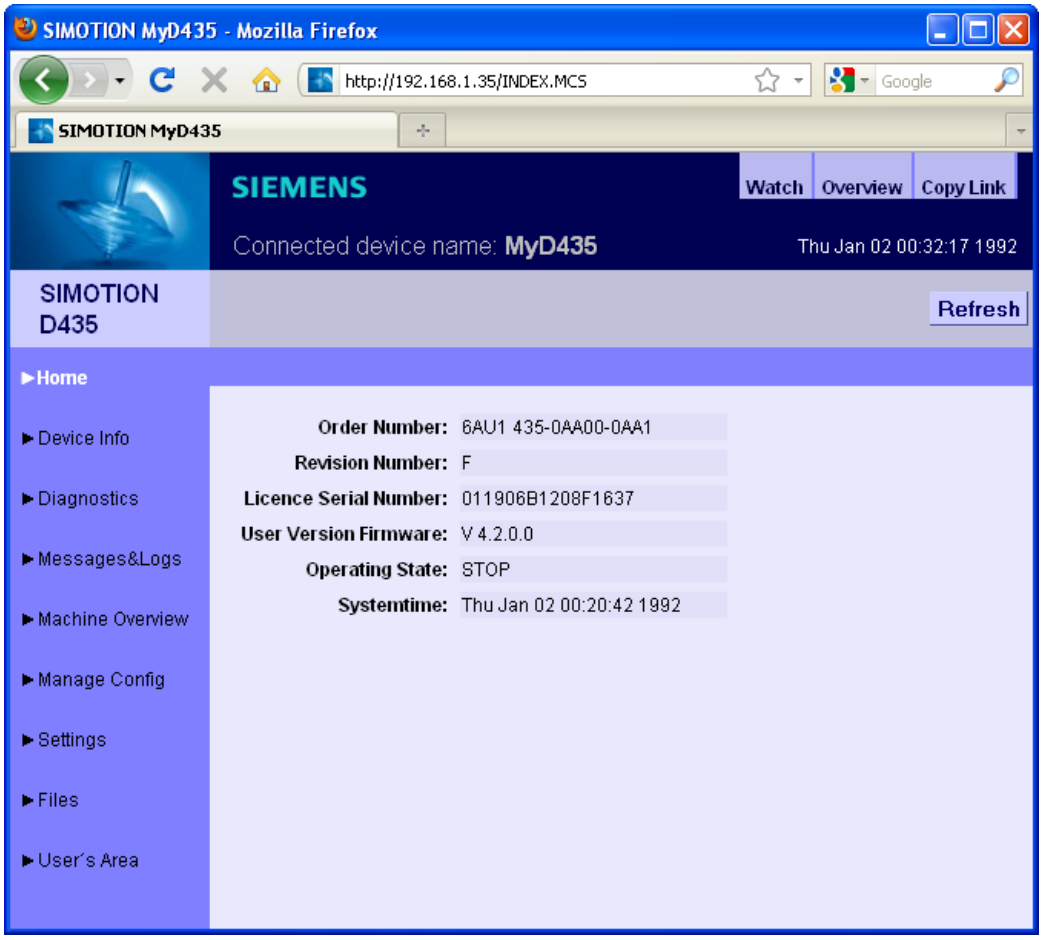


Bild 3-1 Startseite

Weitere Informationen zu den aktuellen Gerätedaten erhalten Sie auf der Seite "Device Info (Seite 26)".

Allgemeine Links

Auf jeder IT DIAG Seite befinden sich drei allgemeine Links:

- "Watch" ermöglicht den Zugriff auf die Watch-Funktion (Seite 35), die in einem separaten Fenster angezeigt wird.
- "Overview" zeigt in einem separaten Fenster den Service Overview (Seite 32) an.
- "CopyLink" ermöglicht es, die URL der aktuellen Seite in die Zwischenablage zu kopieren.

CopyLink

CopyLink zeigt die aktuelle URL der aktuellen Seite.

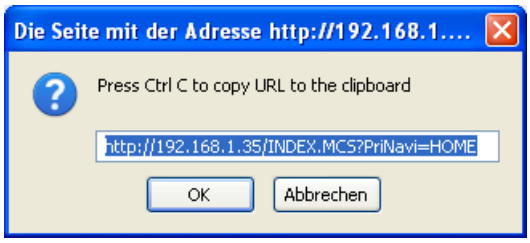


Bild 3-2 CopyLink

3.2.2 Device Info

3.2.2.1 Device Info

Hard- und Firmware-Informationen

Auf der Seite "Device Info" werden folgende aktuelle Hardware- und Firmware-Informationen des SIMOTION Geräts angezeigt:

Manufacturer Name	Siemens AG
Order Number	Liefernummer (MLFB) des Geräts
Revision Number	Hardwareversion
Serial Number	Seriennummer des SIMOTION Geräts
User Version Firmware	SIMOTION Kernel Anwender Version
Build Number	interne Versionsnummer
Additional Hardware	Gesteckte Komponenten des SIMOTION Geräts mit: MLFB, Seriennummer, Revisionsnummer Firmwarenamen, Anwender Versionsnummer, interne Versionsnummer
Technological Packages	Geladene Technologiepakete mit: Paketname, Anwender Versionsnummer, interne Versionsnummer

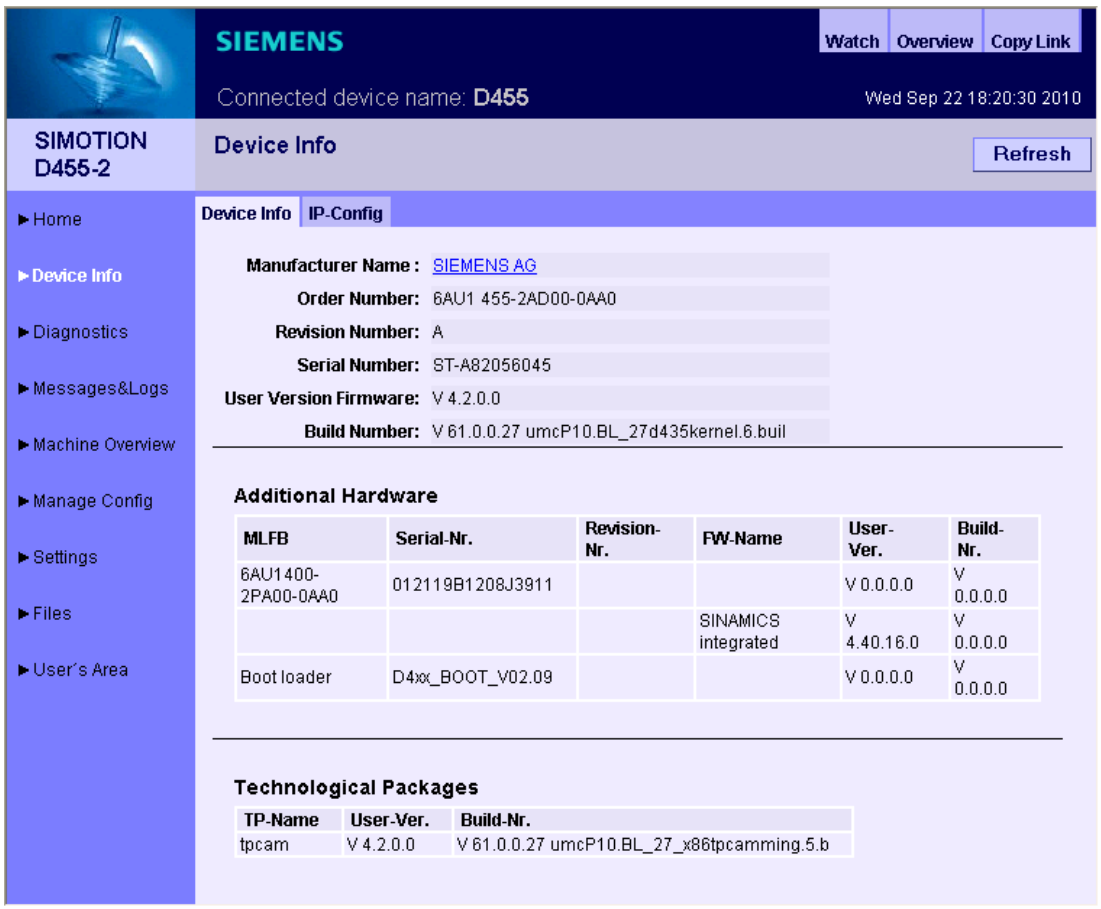


Bild 3-3 Device Info

3.2.2.2 IP-Config

Daten der Ethernet-Schnittstelle des SIMOTION Geräts

Auf der Seite **IP-Config** werden folgende aktuelle Daten der Schnittstelle des SIMOTION Geräts angezeigt:

IP Address	Adresse der TCP/IP Schnittstelle
Subnet Mask	Subnet Mask der Schnittstelle
MAC Address	Adresse der Netzwerkkarte
Gateway	Defaultgateway der Schnittstelle
	Die Angabe steht immer in der ersten Spalte. Sie steht nicht unbedingt im Zusammenhang mit der IP-Adresse der Spalte, sondern kann auch bei den weiteren Schnittstellen projektiert worden sein.
Ethernet-port status:	Übersicht über die Ethernet-Ports. Bei aktiven Ports wird die Geschwindigkeit und die Kommunikationsart des Ports ausgegeben.

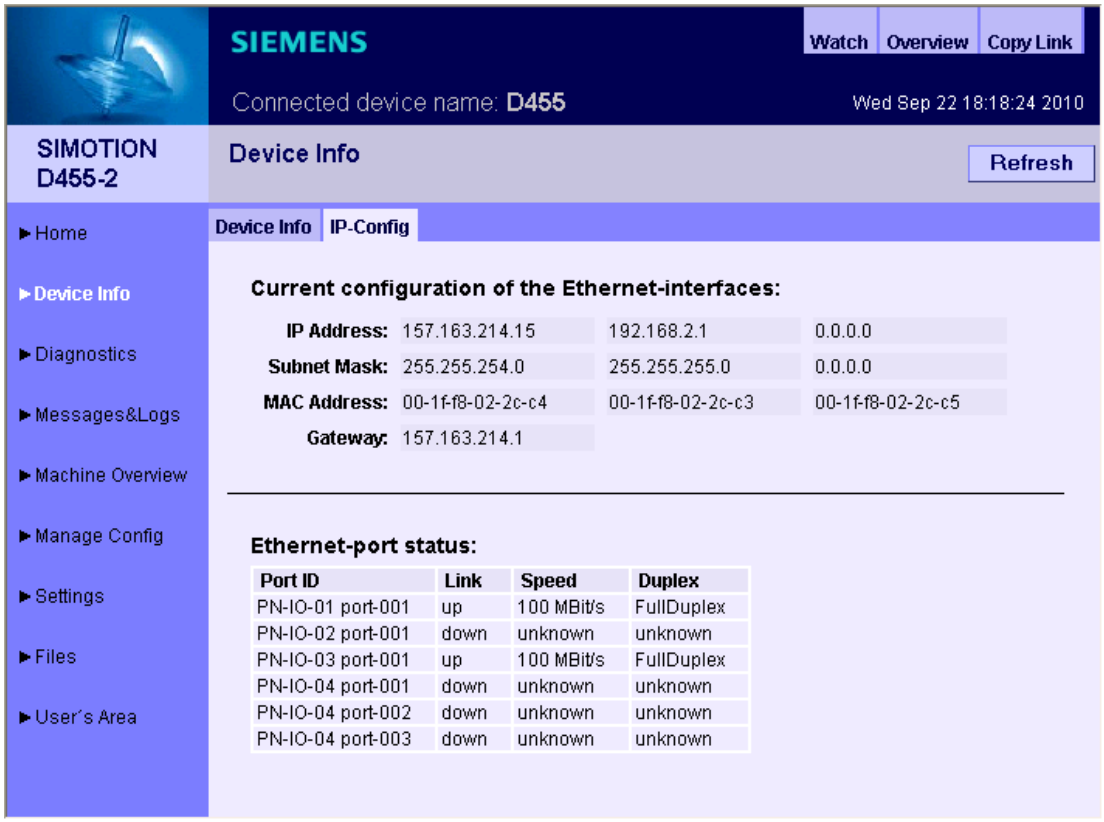


Bild 3-4 IP-Config

3.2.3 Diagnostics

3.2.3.1 Diagnostics

Übersicht über den allgemeinen Zustand des SIMOTION Geräts

Auf der Seite **Diagnostics** werden folgende Zustände des SIMOTION Geräts angezeigt:

Systemtime	Aktuelle Uhrzeit des SIMOTION Geräts
Timezone	Aktuelle Differenz zwischen der Systemtime und der GMT in Minuten
CPU Load by cyclic Tasks	Prozentualer Rechenzeitanteil der Servo und IPO Ebenen an der Gesamtrechenzeit
Memory Load	Größe und Belegung des Speichers, der RAM-Disk, der Speicherkarte und des netzausfesten Speichers in Bytes
State	Aktueller Betriebszustand des SIMOTION Geräts

Über die Reiter der Seite können weitere Details abgefragt werden.

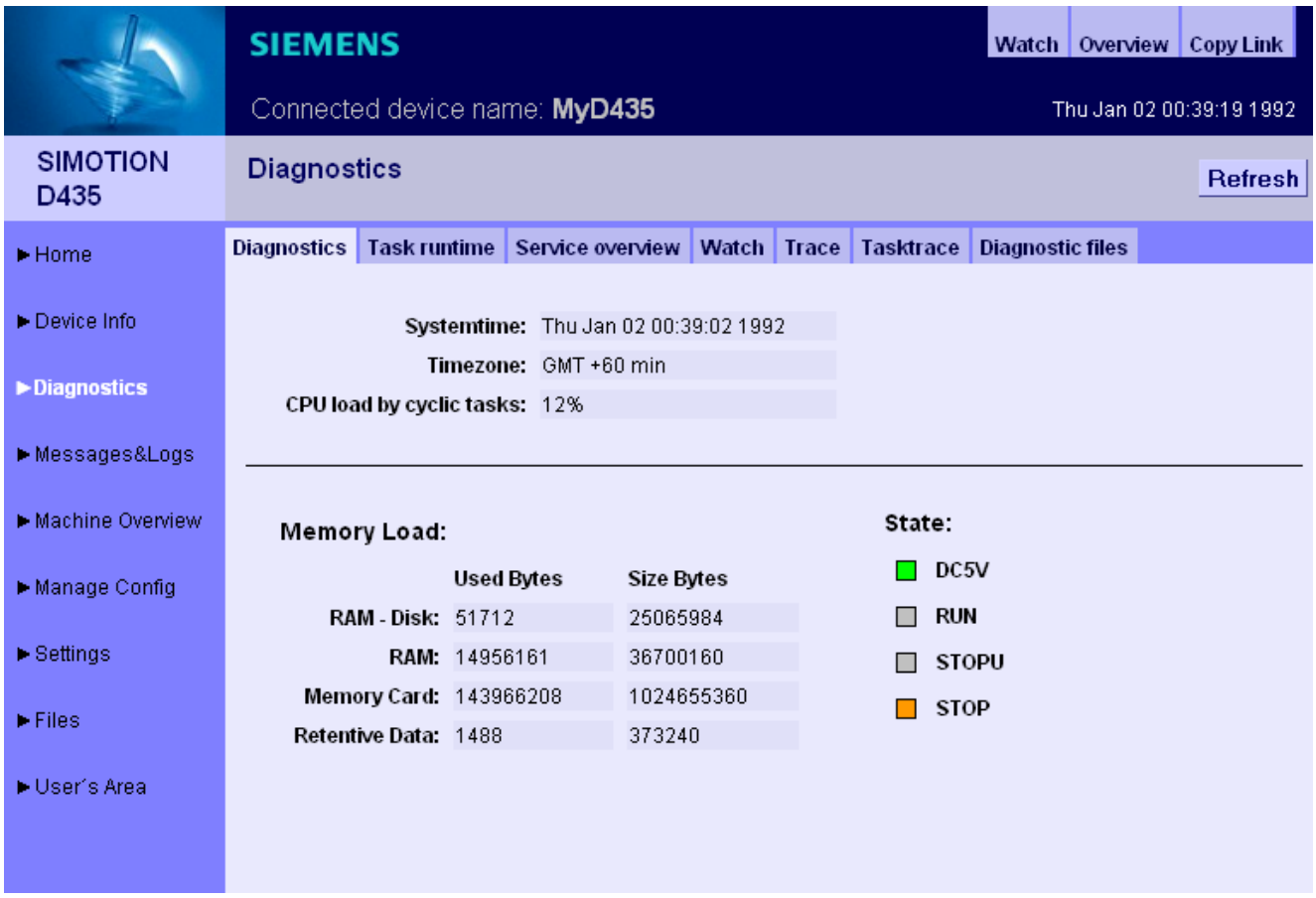


Bild 3-5 Diagnostics

3.2.3.2 Task runtime

Informationen zu Task-Laufzeiten und -Zuständen

Auf der Seite **Task runtime** (zu erreichen über **Diagnostics > Task runtime**) erhalten sie folgende Informationen:

Taskname	Name der Task
Status	Aktueller Status der Task
Actual	Aktuelle Laufzeit der Task in ms
Min	Minimale Laufzeit der Task in ms
Max	Maximale Laufzeit der Task in ms
Average	Mittlere Laufzeit der Task in ms

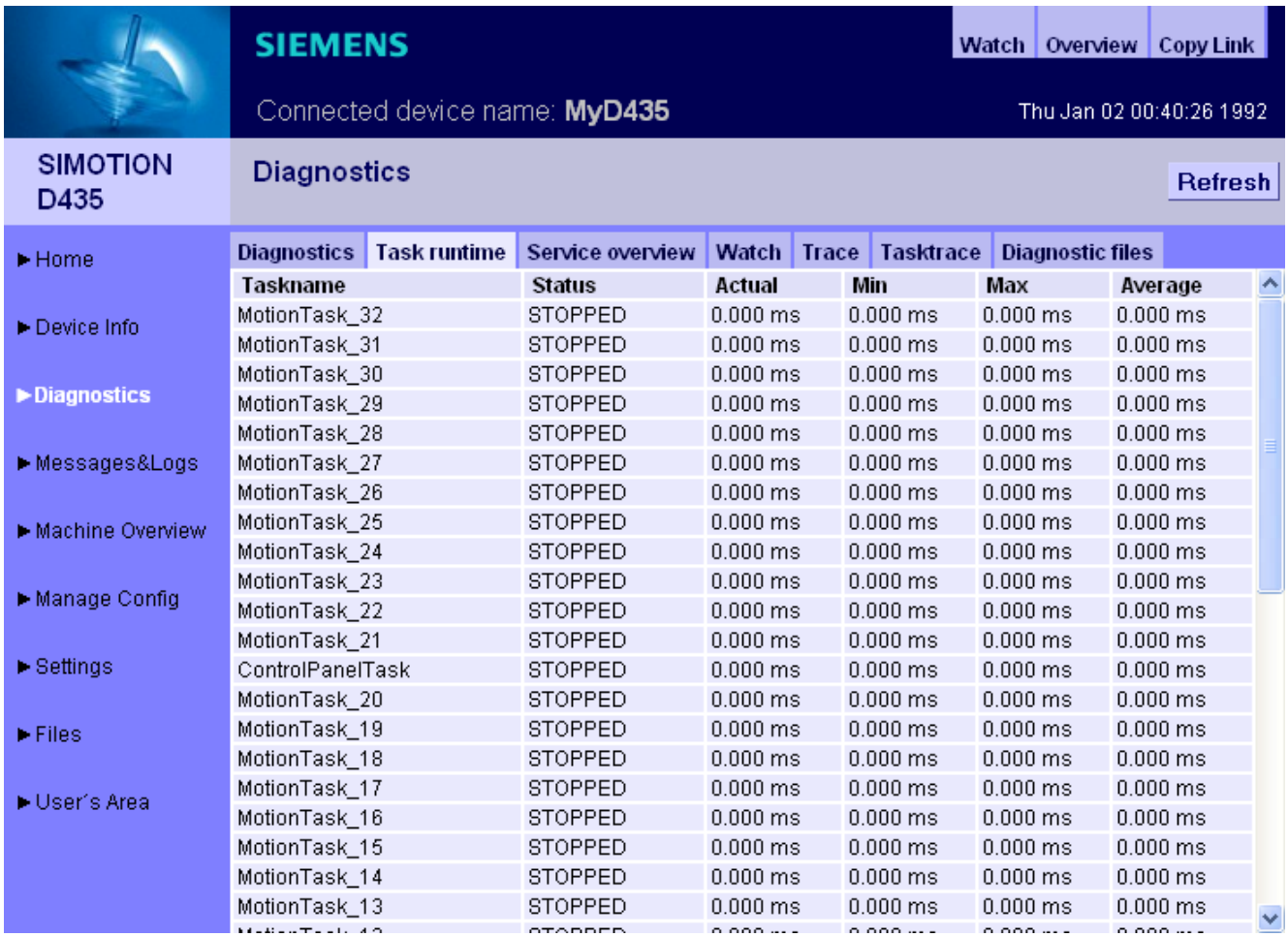


Bild 3-6 Task runtime

3.2.3.3 Service overview

Service overview

Im SIMOTION SCOUT gibt es ein Übersichtsbild, das den Zustand der im Projekt vorhandenen Achsen darstellt. Der Webserver stellt eine entsprechende Seite zur Verfügung.

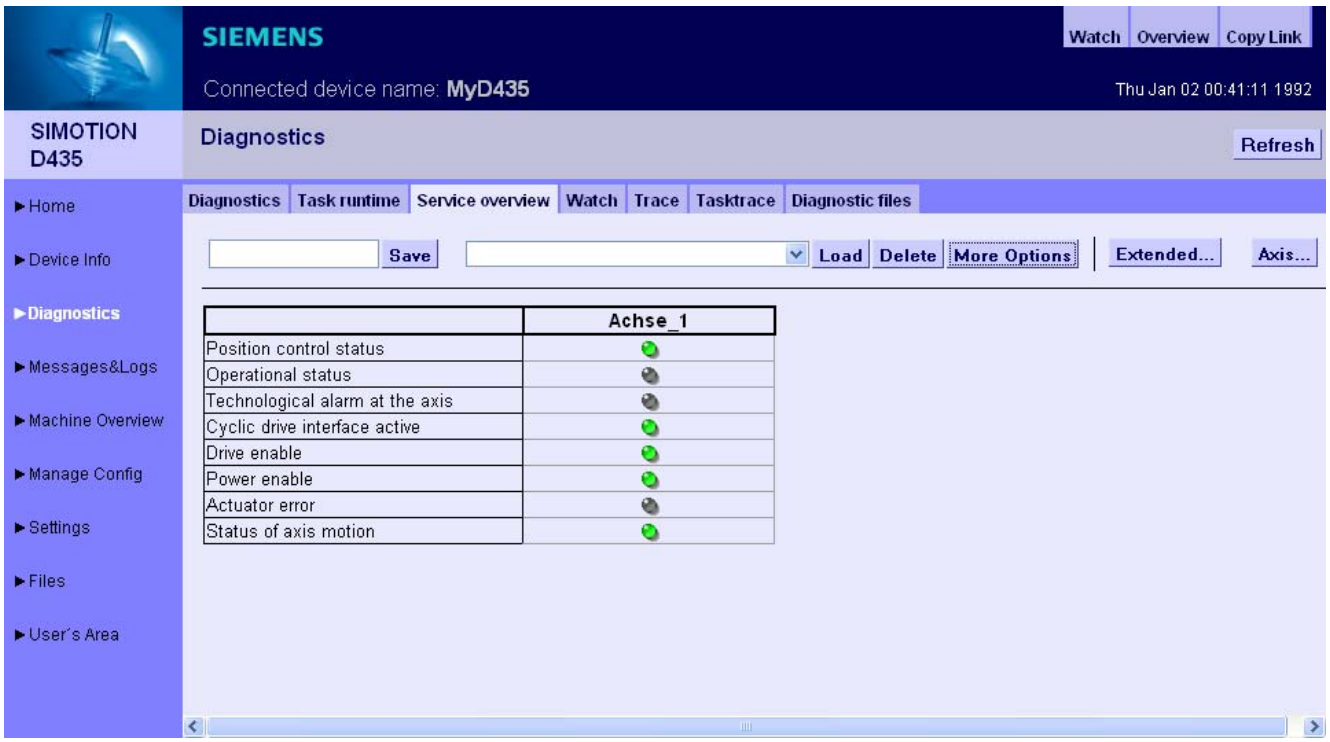


Bild 3-7 Serviceübersicht

In den Spalten der Tabelle werden die Achsen dargestellt. Der **Axis** Button zeigt eine Auswahl aller Achsen und ermöglicht die Auswahl der gewünschten Achsen.

Mit dem **Save** Button kann die aktuelle Einstellung im Gerät gespeichert werden. Im Eingabefeld links vom **Save** Button muss dafür ein Name eingetragen werden.

Mit dem **Load** Button wird eine Einstellung geladen und mit dem **Delete** Button kann sie gelöscht werden.

Der **Extended...** Button öffnet ein Fenster, in dem die gewünschten Systemvariablen ausgewählt werden können.

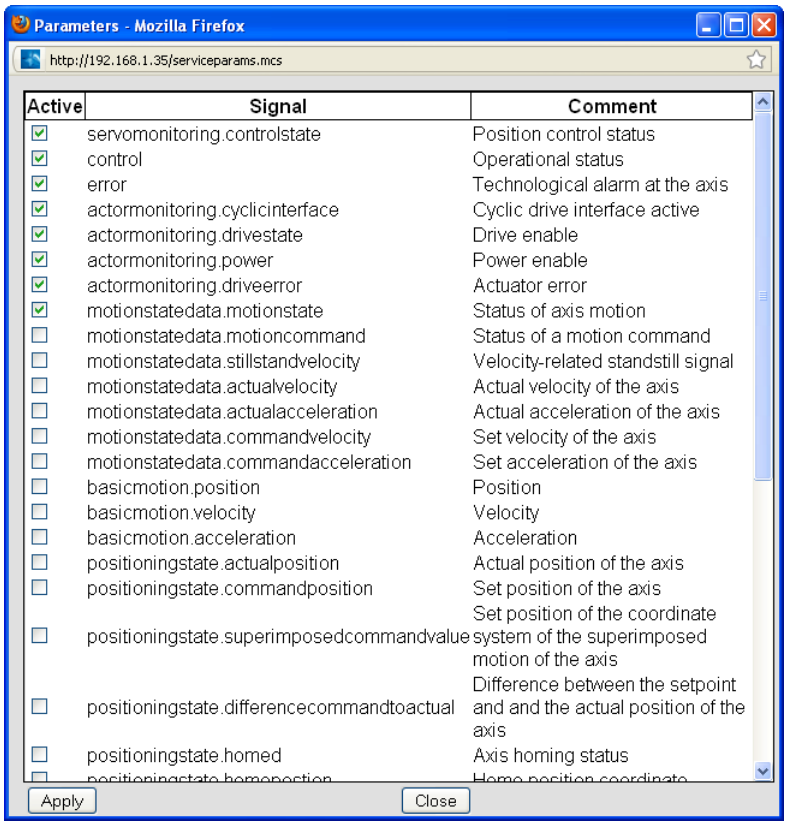


Bild 3-8 Auswahl von Variablen

Eine Auswahl der Achsen eröffnet der **Axis...** Button.

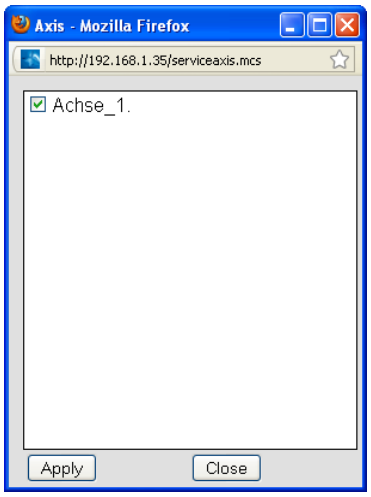


Bild 3-9 Auswahl der Achsen

More Options

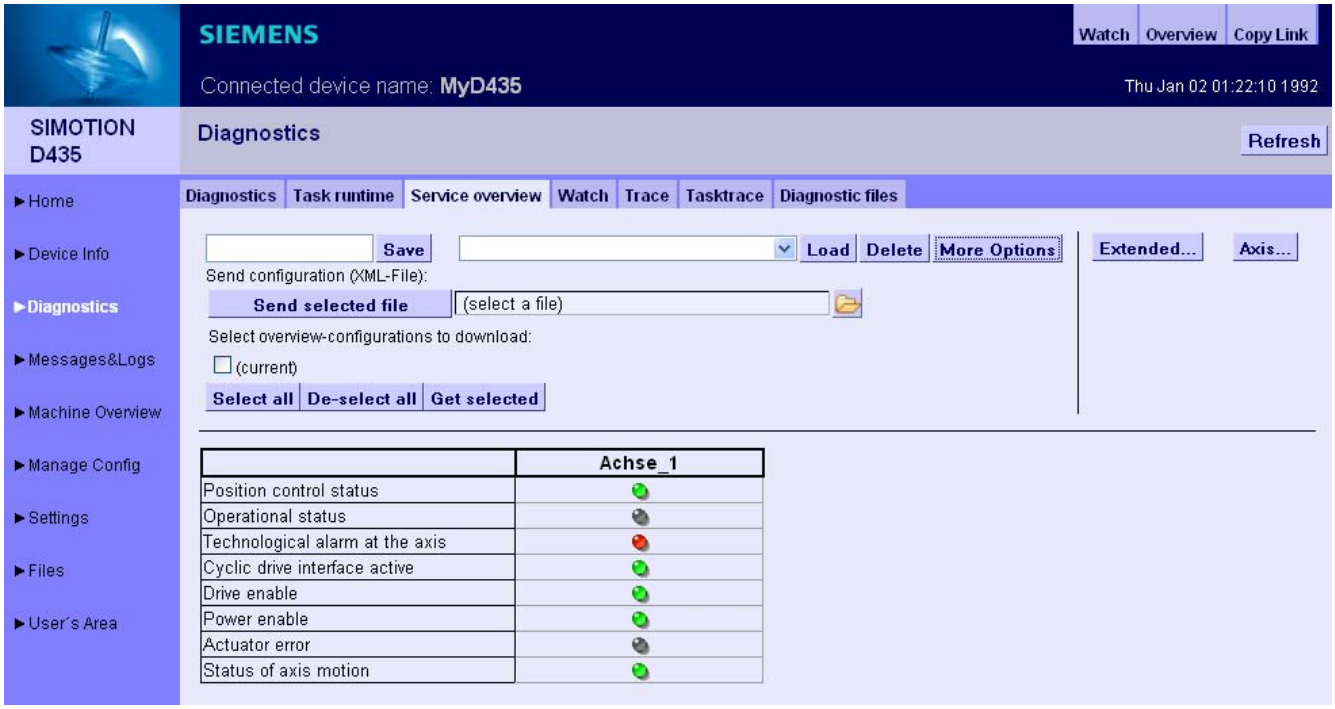


Bild 3-10 Serviceübersicht bei eingeschalteten **More Options**

Der **More Options** Button erweitert den oberen Bildschirmbereich um die Möglichkeit die Konfiguration eines Service overview auf einem PC zu speichern bzw. zu laden.

Der **Send selected file** Button kann eine, z. B. zuvor per E-Mail empfangene, Konfiguration vom PC auf das Gerät geladen werden.

Mittels **Select all** und **De-select all** können alle angezeigten Konfigurationen an- bzw. abgewählt werden.

Mit dem **Get selected** Button können die ausgewählten Konfigurationen als XML-Datei auf dem PC gespeichert werden kann.

3.2.3.4 Watch

Watchtabelle

Diese Seite kombiniert einen Variablenbrowser und eine Watchtabelle. Die Variablen werden mithilfe des Browsers in die Watchtabelle eingetragen.

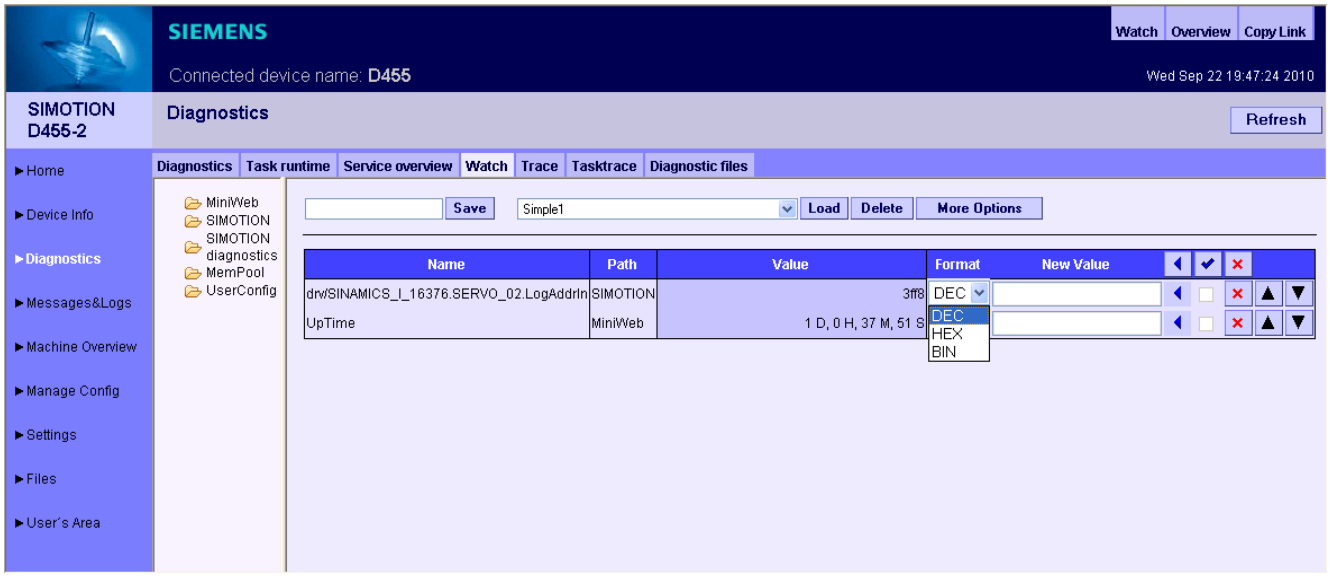


Bild 3-11 Watchtabelle

Für das Beobachten von Variablen stellt der Webserver eine Watchtabelle und einen Symbolbrowser bereit. Der Symbolbrowser bietet die Möglichkeit, den Variablenhaushalt einer SIMOTION Steuerung zu browsen. Diese werden in einer Baumstruktur auf der linken Seite dargestellt. Rechts daneben werden die ausgewählten Variablen dargestellt und können für den Watch bearbeitet werden.

Der Zugriff auf diese Seite ist nur als angemeldeter Benutzer möglich.

Siehe Loginverwaltung (Seite 18)

Die Beobachtung von Unit-Variablen setzt voraus, dass in der jeweils dazugehörigen Unit in den Compilereinstellungen die Option "OPC-XML ermöglichen" aktiviert wurde. Siehe Unit-Variablen verfügbar machen (Seite 214)

Die Format-Spalte gestattet es, bei ganzzahligen Variablen, das Format der Darstellung zu ändern.

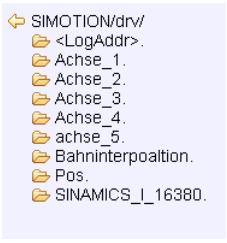
- DEC für die Dezimaldarstellung (Voreinstellung).
- HEX für hexadezimale Darstellung.
- BIN für die binäre Darstellung.

Alle eingegebenen Steuerwerte werden entsprechend dieser Einstellung interpretiert.

Zugriff auf die Antriebsparameter

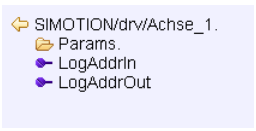
Der Zugriff auf die Antriebsparameter erfolgt über eine Baumstruktur. Die Auswahl entspricht dem Zugriff auf Variablen über den "Variablen Provider". Siehe Variablen Provider (Seite 239)

Parameter werden als Zahl ohne führendes 'p' oder 'r' angezeigt. Der Parameter r0002 wird z. B. zu 0002.

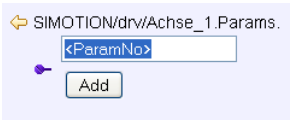


Es existieren drei Möglichkeiten, um auf die Antriebsparameter zuzugreifen:

1. Technologieobjekt Achse

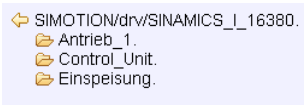


Auswahl eines Technologieobjekts

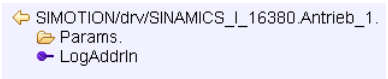


Auswahl eines Antriebsparameters

2. Driveobjekt Adressierung



Auswahl eines Driveobjekts (der Name wird aus der Diagnoseadresse erzeugt)



Auswahl eines Antriebs

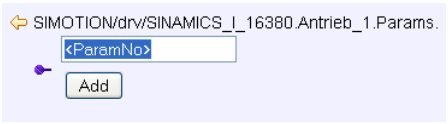


Bild 3-12 Auswahl eines DO Parameter

3. Logische Adresse

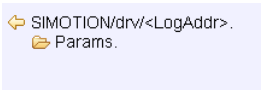
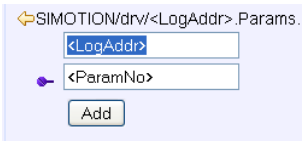


Bild 3-13 Auswahl einer logischen Adresse



Auswahl eines Antriebsparameters und einer logischen Adresse

Bearbeitung der Watchtabelle

Die Tabelle kann mit dem **Save** Button gespeichert werden. Dazu ist es erforderlich, im Eingabefeld neben dem Button einen Namen einzugeben.

Mit dem **Load** Button kann eine gespeicherte Tabelle geladen und mit dem **Delete** Button gelöscht werden.

Mit dem Lösch-Button (rotes Kreuz) in der Titelzeile der Tabelle können alle Tabellenzeilen auf einmal gelöscht werden. Einzelne Tabellenzeilen können mit den entsprechenden Buttons am Ende der jeweiligen Zeile gelöscht werden.

Durch das Drücken des **More Options** Button wird der obere Bildschirmbereich um die Möglichkeit erweitert, die Einstellungen der Watchtabellen auf einem PC zu speichern und später wieder in die Steuerung zu laden.

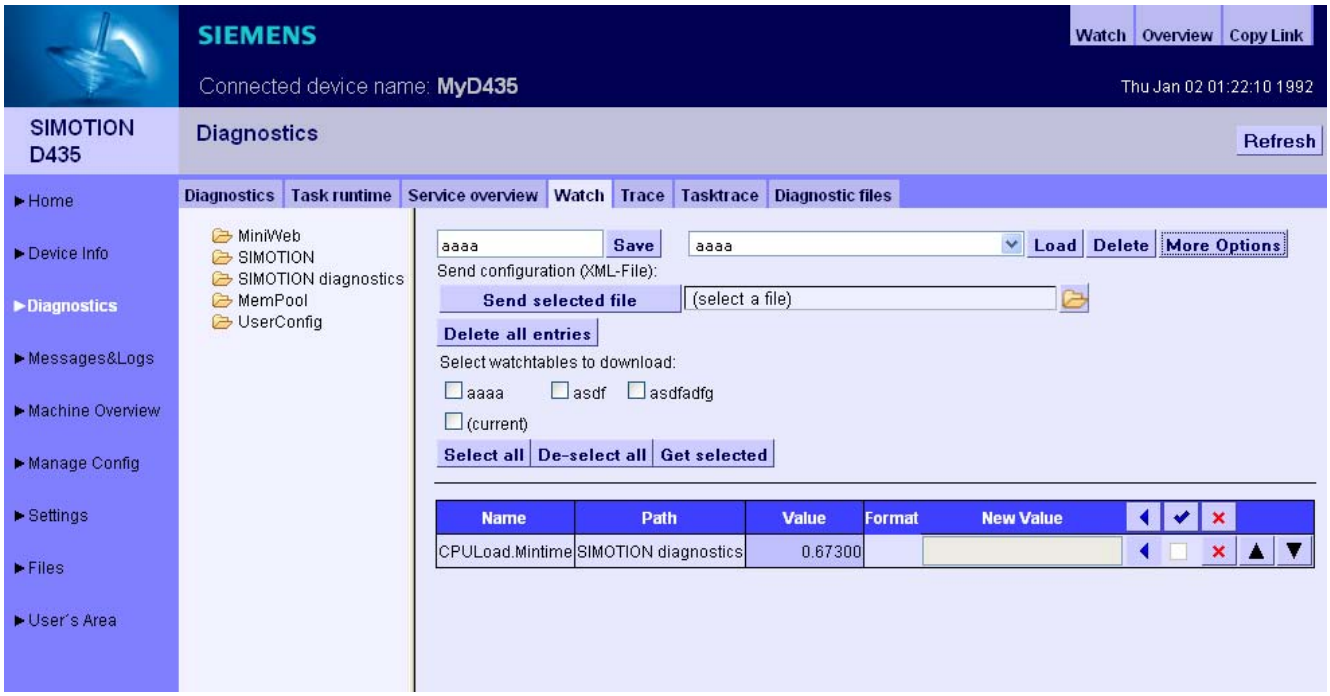


Bild 3-14 Watch More Options

Eine ausführlichere Beschreibung der Funktionalität der **More Options** findet sich im Abschnitt Service Overview.

Siehe auch

Service overview (Seite 32)

3.2.3.5 Trace (Gerätetrace)

Einrichten eines Gerätetrace (Device Trace)

Die SIMOTION Steuerung stellt dem Anwender die Möglichkeit, einen Variablentrace einzurichten, über einen Webservice zur Verfügung.

Ab Version 4.2 wird neben dem hier beschriebenen Gerätetrace noch ein verteilter Trace (Seite 43) (System Trace) bereitgestellt.

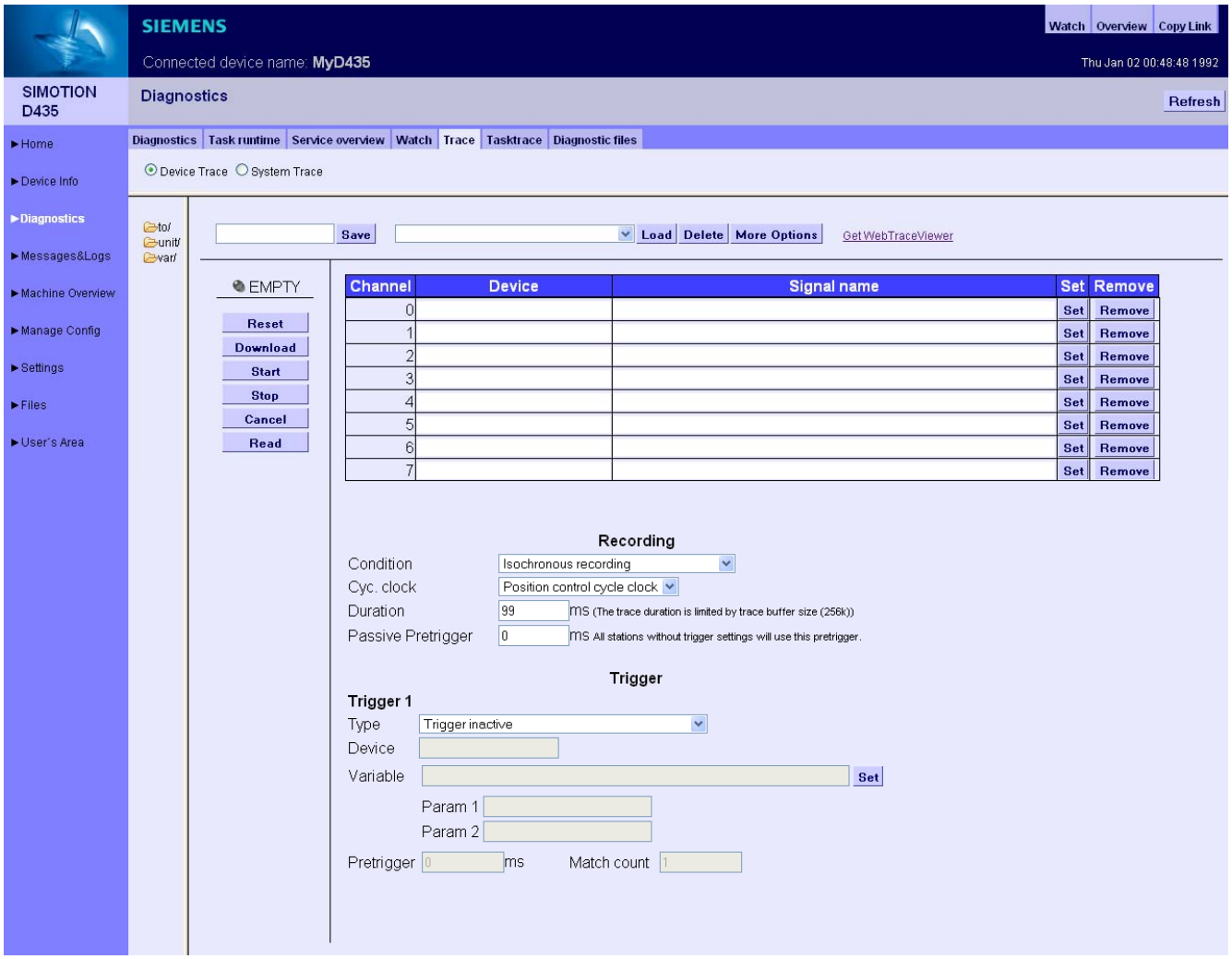


Bild 3-15 Gerätetrace

Vorgehensweise zur Erstellung und Ausführung eines Gerätetrace:

- Schaltknopf **Device Trace** wählen
- Auswählen des gewünschten Signals aus Liste der Provider (to, unit oder var)
- Markiertes Symbol wird durch Betätigen des **Set** Button auf gewünschtes Signal gesetzt
- Aufzeichnungs- und Triggerbedingungen setzten
- **Download** – Laden der Einstellungen auf die Steuerung
- **Start** – Starten des Trace
- **Stop** – Stoppen des Trace (nur bei Manuellem Trace notwendig)
- **Read** – Laden der Trace-Ergebnisse als WTRC-Datei auf den PC. Die WTRC-Datei wird dadurch auf dem Gerät gelöscht.
- Betrachten der WTRC-Datei mit WebTraceViewer
- **Cancel** – Löschen der Einstellungen auf der Steuerung
- **Reset** – Löschen der Einstellungen auf der Webseite

Mit dem **Read** Button wird eine Datei mit der Endung WTRC erzeugt, die die aktuellen Trace-Daten enthält. Die Datei kann gespeichert oder mit dem Programm WebTraceViewer betrachtet werden.

Durch das Drücken des **More Options** Button wird der obere Bildschirmbereich um die Möglichkeit erweitert, die Einstellungen des Gerätetrace auf einem PC zu speichern und später wieder in die Steuerung zu laden.

Der Zugriff auf diese Seite ist nur als angemeldeter Benutzer möglich. Siehe Loginverwaltung (Seite 18)

Hinweis

Für den Trace steht nur ein begrenzter Speicherplatz von 256 KB zur Verfügung, der als Ringpfuffer organisiert ist.

Trace Modi

Der Gerätetrace kann in zwei Modi betrieben werden:

1. Getriggert
   Der Trace startet nach Eintreffen eines Triggerereignisses und wird nach Ablauf einer parametrierbaren Zeit oder bei vollem Tracebuffer gestoppt.

2. Endlos
   Der Trace startet sofort und läuft so lange, bis er per Bedienhandlung gestoppt wird. Die Tracedaten werden von einem Client schritthaltend abgeholt.

Eine Beschreibung der Aufzeichnungseinstellungen und Triggerbedingungen finden Sie im Abschnitt Systemtrace.

Speicherung und Laden einer Trace Konfiguration

Mit dem **Save** Button kann eine Konfiguration unter einem Namen auf dem Gerät gespeichert und mit dem **Load** Button wieder geladen werden. Eine ausführlichere Beschreibung der Funktionalität der **More Options** findet sich im Abschnitt Service Overview (Seite 32).

WebTraceViewer

Für die Darstellung der Trace-Daten steht das PC-Programm WebTraceViewer zur Verfügung. Über den Link **GetWebTraceViewer** kann der WebTraceViewer auf dem PC gespeichert kann. Bei den C-Baugruppen steht dieser Link nicht zur Verfügung. Alternativ kann der WebTraceViewer über den E-Support bezogen oder von der Addon DVD kopiert werden.

Das Programm ist in der Lage, die in einer WTRC-Datei gespeicherten Daten, grafisch darzustellen.

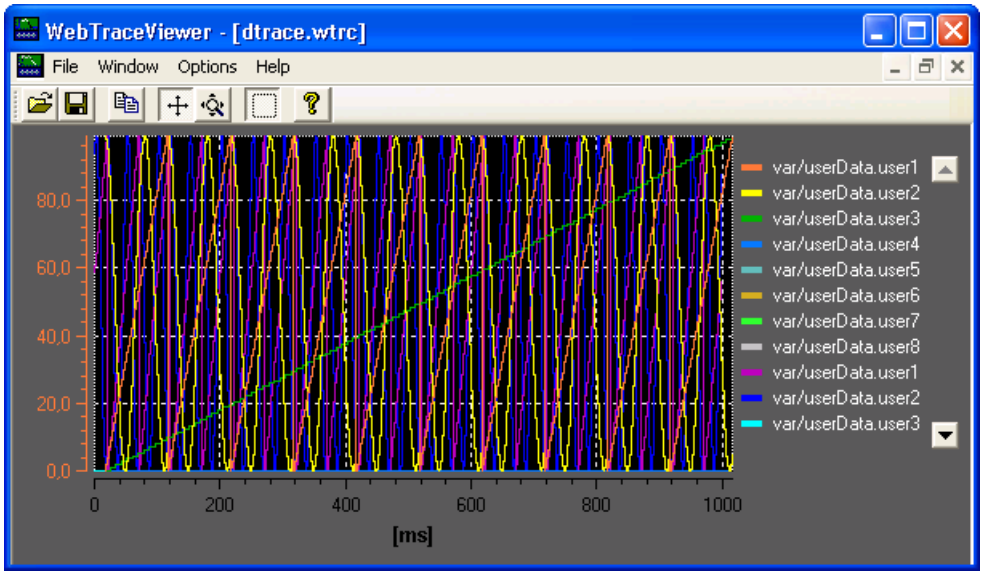


Bild 3-16 WebTraceViewer

Funktionen der Buttons

1. Datei öffnen: Ermöglicht das Öffnen von WTRC-Dateien.
2. Datei speichern: Ermöglicht die Speicherung von WTRC-Dateien.
3. Kopieren: Kopiert den Inhalt des aktuellen WTRC-Fensters als Bitmap in die Zwischenablage. So kann die Grafik z. B. in eine Textverarbeitung kopiert werden.
4. Scroll Modus: Ermöglicht die Verschiebung des sichtbaren Bereichs der Grafik mit der Maus.
5. Zoom Modus: Ermöglicht die Dehnung und Stauchung der Grafik mit der Maus.
6. Auswahl Modus: Wenn dieser Button gedrückt ist, kann nur noch ein rechteckiger Bereich der Grafik ausgewählt werden. Die Buttons 4. und 5. haben dann keine Bedeutung mehr.

CSV Export

Der Menüpunkt **File Export** bietet die Möglichkeit die Trace-Daten im CSV-Format zu speichern, um sie z. B. in einer Tabellenkalkulation einzulesen.

Defekte WTRC-Dateien

Wenn der WebTraceViewer eine defekte Datei einliest, gibt er einen Hinweis auf den Fehler aus.

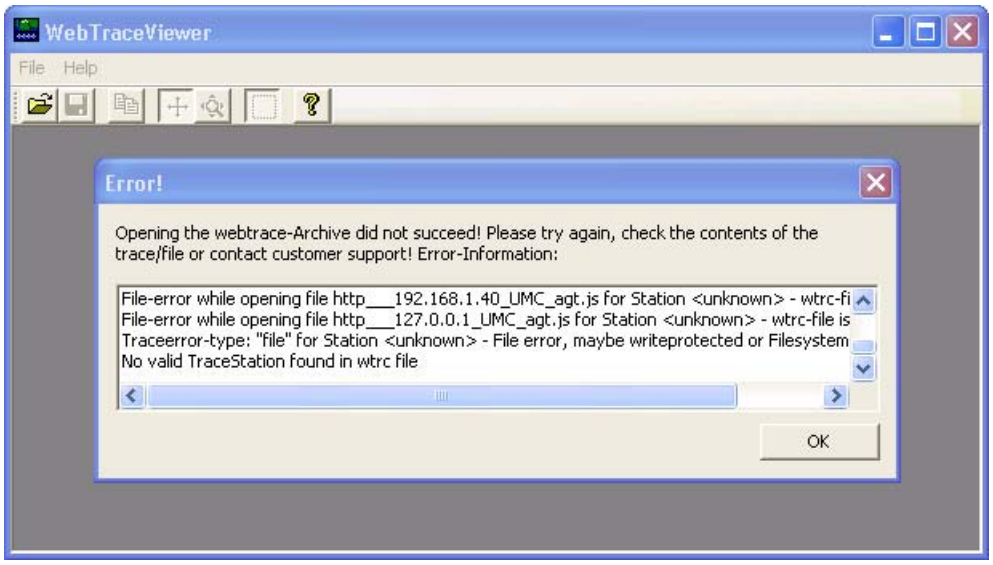


Bild 3-17 WebTraceViewer mit fehlerhafter WTRC-Datei

3.2.3.6 Trace (Systemtrace)

Systemtrace einrichten und ausführen

Der Systemtrace steht ab der SIMOTION-Version 4.2 zur Verfügung. Mit dem Systemtrace kann ein Trace über mehrere Geräte durchgeführt werden.

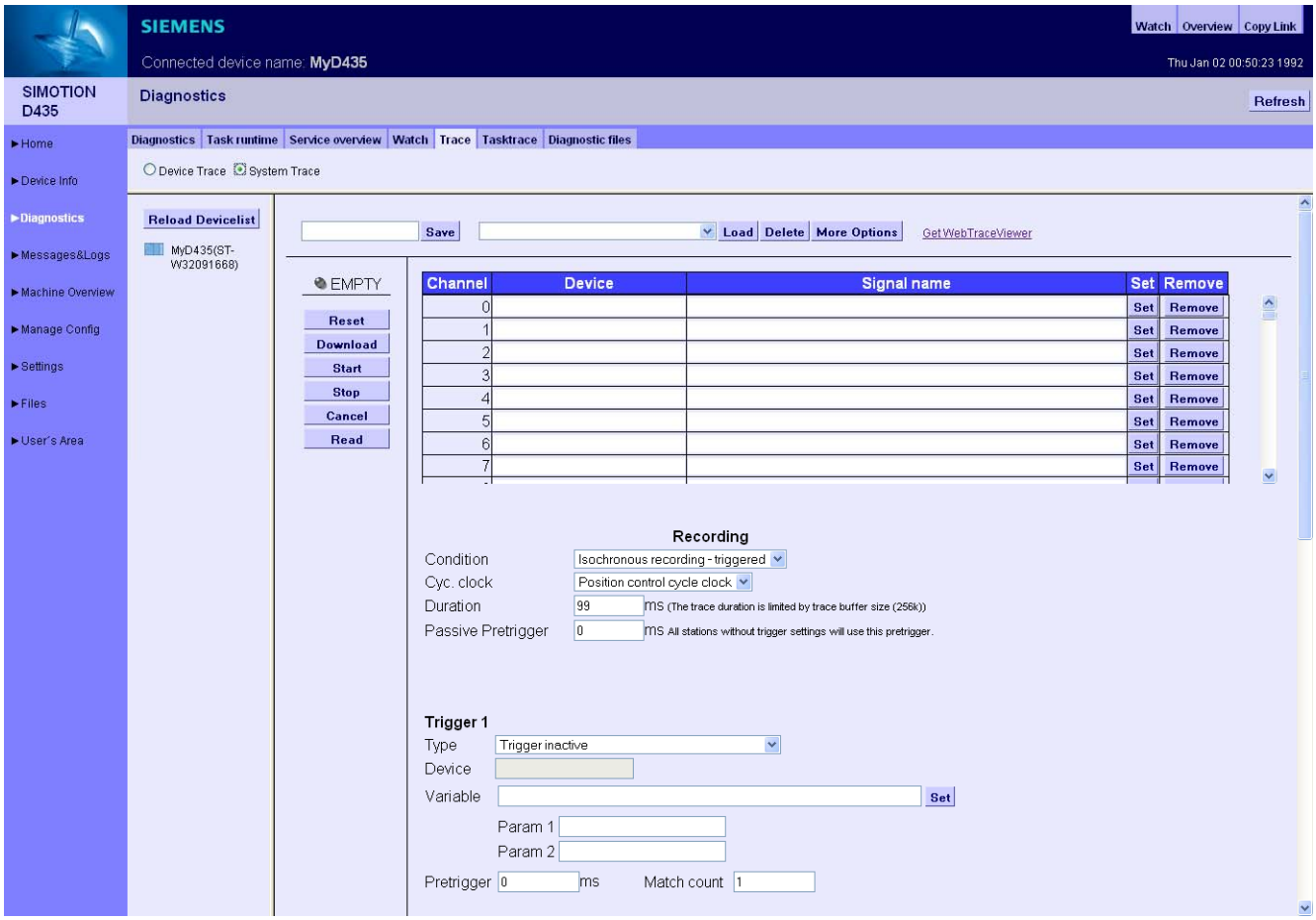


Bild 3-18 Systemtrace 1

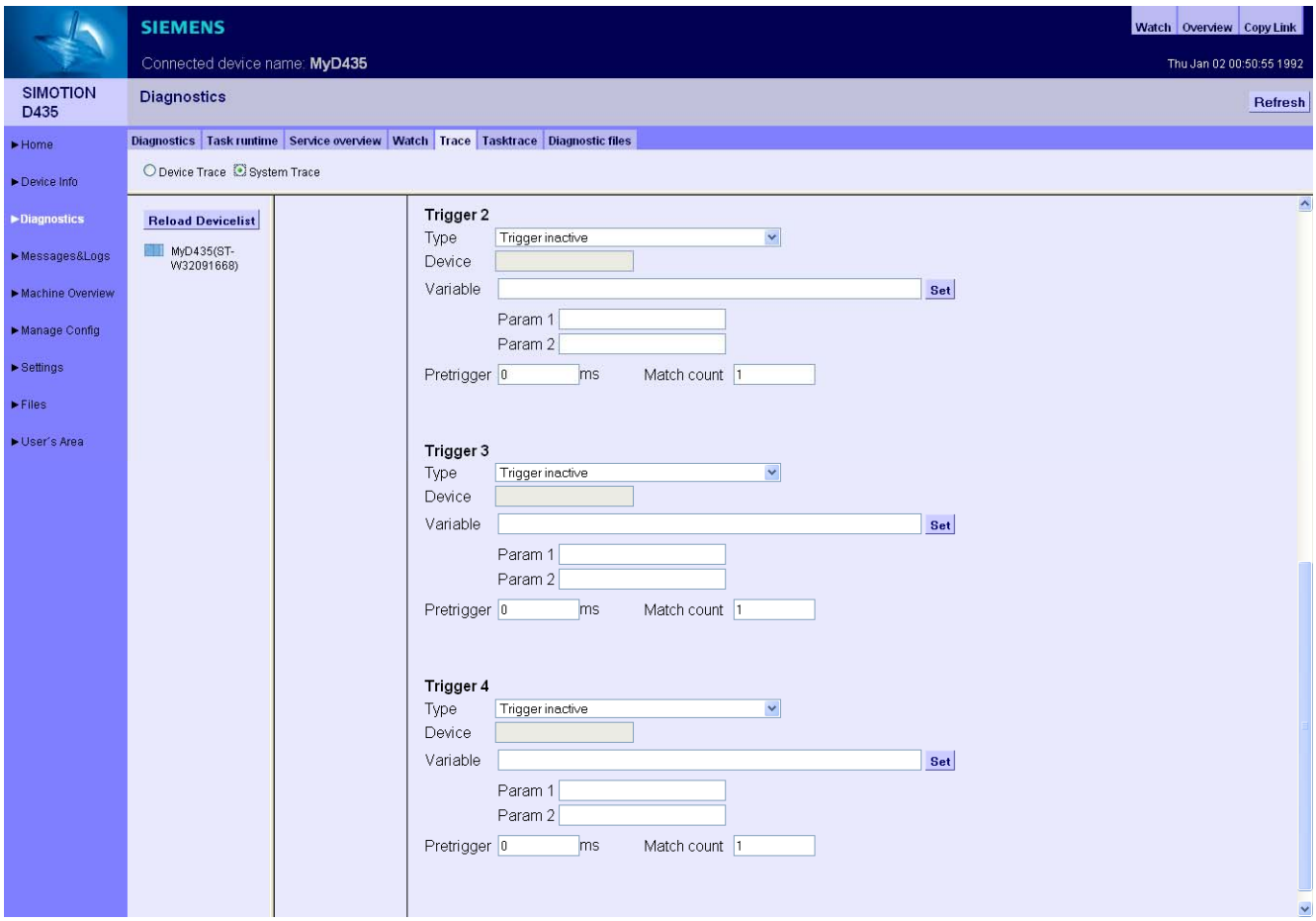


Bild 3-19 Systemtrace 2

Vorgehensweise zur Erstellung und Ausführung eines Systemtrace:

- Schaltknopf **System Trace** wählen
- Auswählen des gewünschten Signals aus der Geräteliste der Provider (to, unit oder var)
- Markiertes Symbol wird durch Betätigen des **Set** Button auf gewünschtes Signal gesetzt
- Aufzeichnungs- und Triggerbedingungen setzten
- **Download** – Laden der Einstellungen auf die Steuerung
- **Start** – Starten des Systemtrace
- **Stop** – Stoppen des Systemtrace (nur bei Manuellem Trace notwendig)
- **Read** – Laden der Traceergebnisse als WTRC-Datei auf den PC. Die WTRC-Datei wird dadurch auf dem Gerät gelöscht.
- Betrachten der WTRC-Datei mit dem PC-Programm WebTraceViewer
- **Cancel** – Löschen der Einstellungen auf der Steuerung
- **Reset** – Löschen der Einstellungen auf der Webseite

Voraussetzungen

Für die Zeitsynchronisation des verteilten Trace müssen die Geräte über PROFINET IO IRT verbunden und synchronisiert sein. Der Trace kann auf beliebig viele Geräte angewendet werden, allerdings ist die maximale Anzahl der Signale auf 128 begrenzt, wovon maximal 8 pro Gerät eingesetzt werden können. Pro Gerät ist maximal ein Trigger möglich.

Nach der Auswahl der Signale muss eine Zuordnung der gewünschten Aufzeichnungs- und Triggerbedingung stattfinden.

Trace Modi

Der Systemtrace kann nur im Modus 'Getriggert' betrieben werden. Der Trace startet nach Eintreffen eines Triggerereignisses und wird nach Ablauf einer parametrierbaren Zeit oder bei vollem Tracebuffer gestoppt.

Aufzeichnungseinstellungen

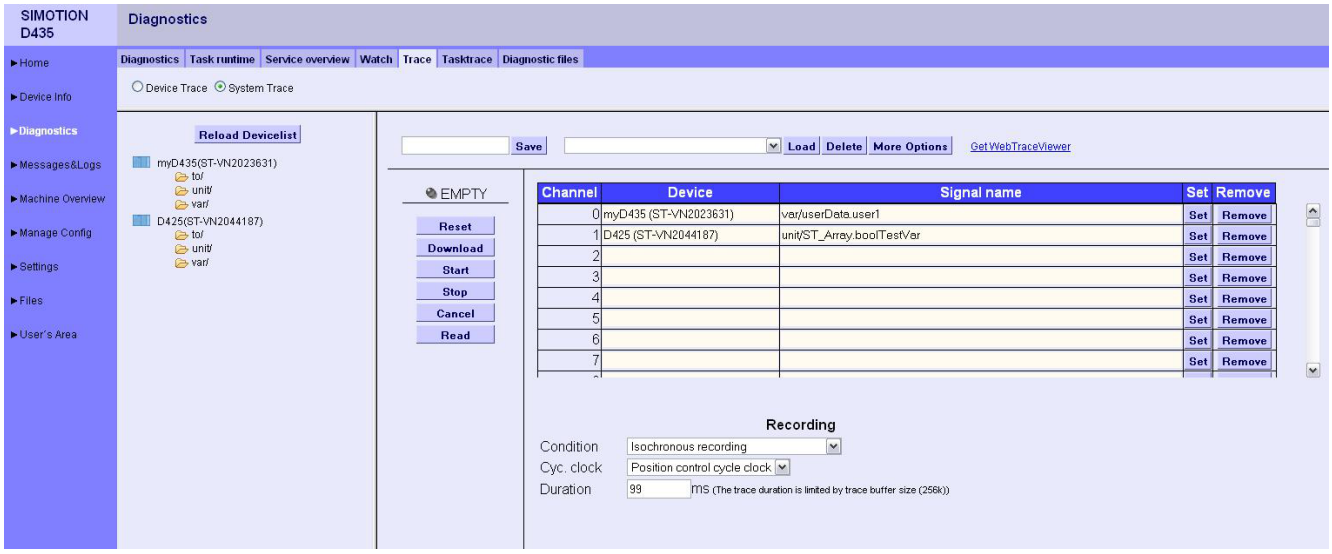


Bild 3-20 Beispiel: Recording Einstellungen

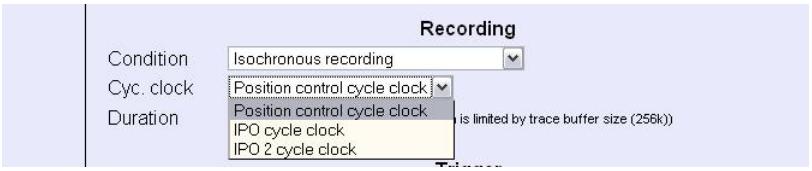


Bild 3-21 Beispiel: Basistakt Einstellung

- Condition: Messwerterfassung
- Cyc. Clock: Basistakt
- Aufzeichnung in Ringpuffer mit max. 256k Größe

Triggerbedingungen

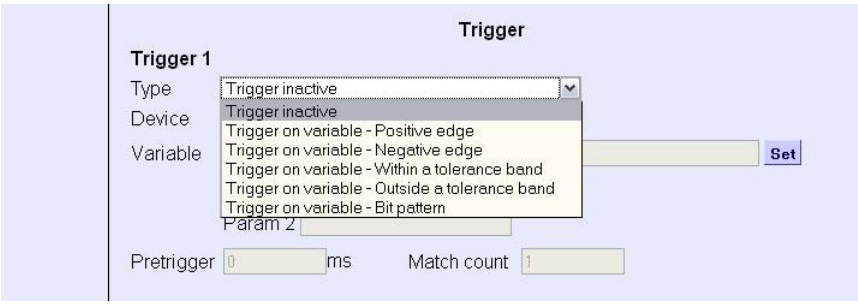


Bild 3-22 Beispiel: Trigger Einstellung

Bezeichnung	Erklärung	Operand 1	Operand 2
Positive Edge	Steigende Flanke löst aus, wenn Variable größer wird	-	-
Negative Edge	Fallende Flanke löst aus, wenn Variable kleiner wird	-	-
Within a tolerance band	Innerhalb eines Wertebereichs löst aus, wenn Variable sich innerhalb des angegebenen Intervalls befindet	Untere Grenze des Intervalls	Obere Grenze des Interfalls
Outside a tolerance band	Außerhalb eines Wertebereichs löst aus, wenn Variable sich außerhalb des angegebenen Intervalls befindet	Untere Grenze des Intervalls	Obere Grenze des Interfalls
Bit pattern	Das Bitmuster löst aus, wenn die Variable verundet mit Operand1 ungleich null ist	Bitmuster	-

Übersicht Triggerbedingungen

Trigger

- Steigende Flanke, fallende Flanke
- Pretrigger = Zeit in ms, beim Schalten des Triggers gehört dieser "Vorlauf" mit zur Aufzeichnung

Initialisierung

Zur Initialisierung des Trace werden die Tracevariablen und Triggerbedingungen an die beteiligten Geräte übertragen. Wenn die Initialisierung zumindest an einem Gerät fehlerfrei abgeschlossen wurde, kann der Trace gestartet werden.

Betrachten des Trace

Mit dem PC-Programm WebTraceViewer können die Trace-Daten auf dem PC angezeigt werden.

Siehe auch

Trace (Gerätetrace) (Seite 39)

3.2.3.7 Tasktrace

Tasktrace

Diese Seite ermöglicht die Einrichtung und Steuerung des SIMOTION Tasktrace (inkl. Triggerbedingungen).

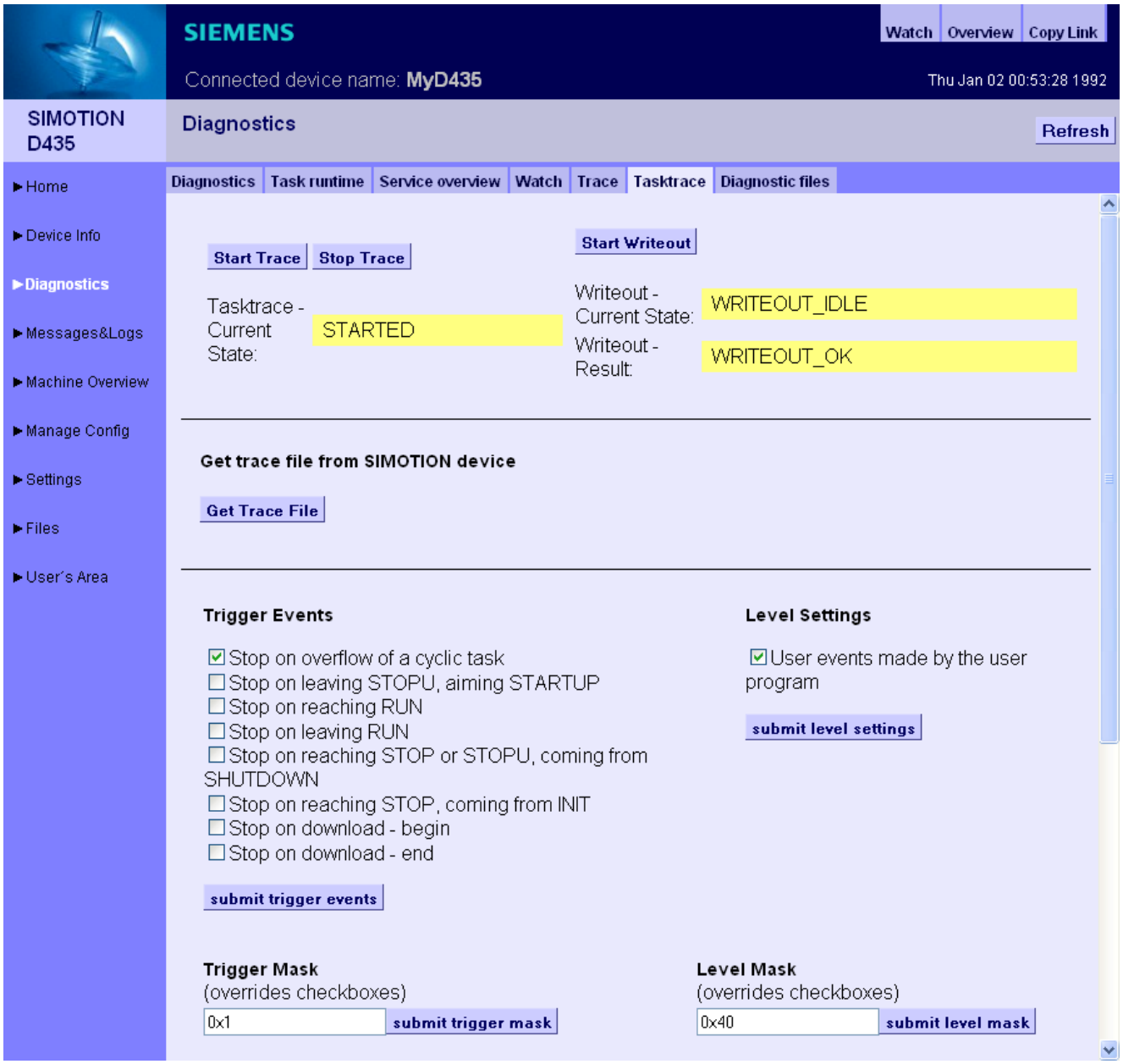


Bild 3-23 Tasktrace oberer Bereich

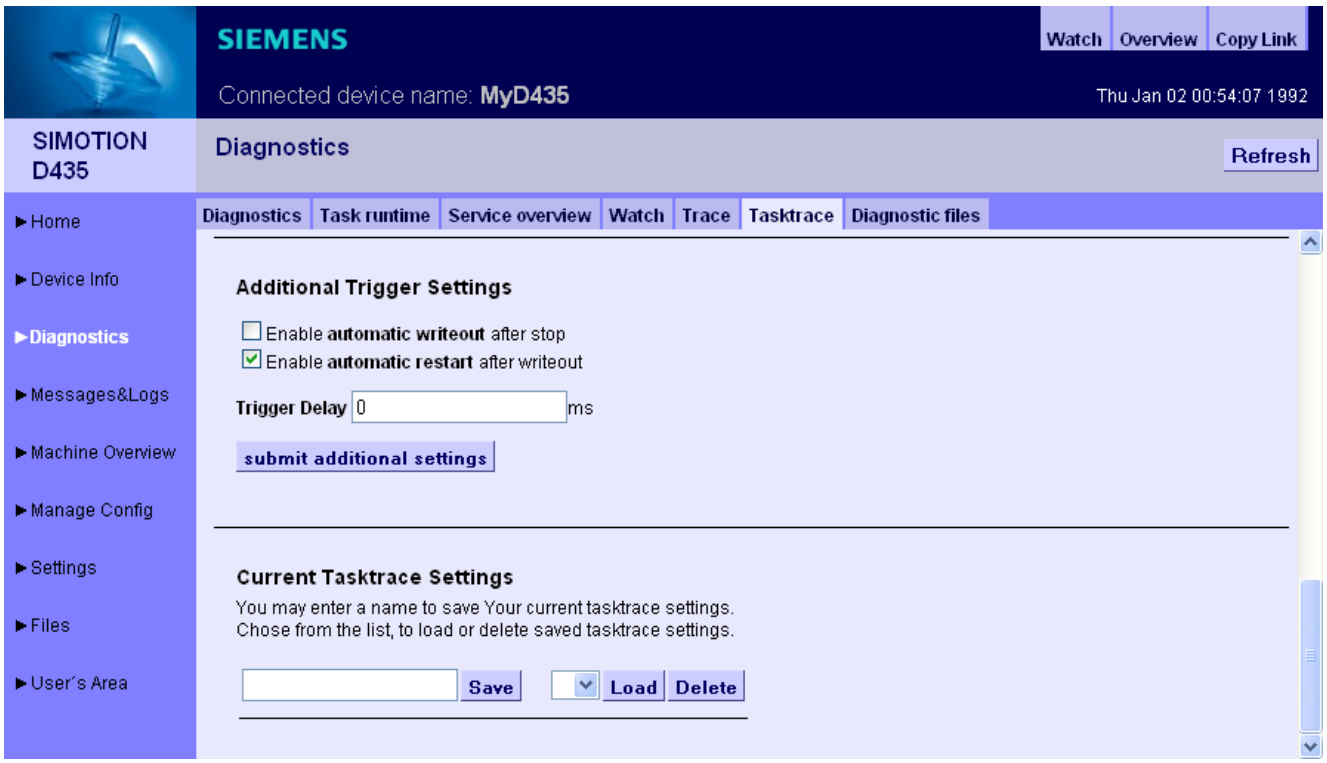


Bild 3-24 Tasktrace unterer Bereich

Der Tasktrace stellt eine Diagnosemöglichkeit zur Laufzeit dar, mit deren Hilfe Rückschlüsse auf die Vorgänge in den einzelnen Tasks getroffen werden können (z. B. Taskwechsel).

Die Traceaufzeichnung wird fortlaufend in einen Ringpuffer geschrieben.

Einmal gestartet kann eine Traceaufzeichnung manuell gestoppt oder durch ein Triggerereignis bedingt angehalten werden. Anschließend kann die Aufzeichnung durch betätigen des Buttons **Get Trace File** auf den PC geladen und mit dem Task Profiler angezeigt werden.

Start Trace

Der Button **Start Trace** startet den Tasktrace mit den zuvor gemachten und an das Gerät per **Submit** übertragenen Einstellungen.

Stop Trace

Mit dem **Stop Trace** Button kann der Trace manuell gestoppt werden.

Der Zustand des Trace wird im Feld **Tasktrace - Current State:** angezeigt.

Start Writeout

Der **Start Writeout** Button schreibt den Inhalt des Tracepuffer in die Datei "/USER/SIMOTION/SYSLOG/TASKTRACE/TTRACE.JEN" auf dem Gerät.

Der Zustand des Schreibvorgangs wird in den Feldern **Writeout - Current State:** und **Writeout - Result:** angezeigt.

Get Trace File

Mit dem Button **Get Trace File** kann die Datei TTrace.jen auf den PC geladen und mit dem Programm TaskProfiler angezeigt werden. Das Setup des TaskProfiler befindet sich auf der Installations-DVD im Addon-Verzeichnis.

Die Benutzung dieses Programms setzt die Java Runtime ab Version 1.6 voraus.

Trigger Events

Die **Trigger Events** können über verschiedene Checkboxen ausgewählt und beliebig kombiniert werden. Der Button **submit trigger events** übertragt die Auswahl auf das Gerät.

Trigger Mask

Das Eingabefeld **Trigger Mask** ermöglicht es dem Experten **Trigger Events** als kodierte Zahl einzugeben. Der Button **submit trigger mask** überträgt die Eingabe an das Gerät und überschreibt alle vorher gemachten Eingaben.

Level Settings / Level Mask

Mithilfe dieser Einstellungen kann eingestellt werden, welche Ereignisse in den Tasktrace eingetragen werden.

Additional Trigger Settings

Diese Einstellungen ermöglichen es einen Trace automatisch zu sichern.

- **Enable automatic writeout after stop**: Nach dem Eintreten eines Triggerereignisses werden die Tracedaten automatisch gesichert.
- **Enable automatic restart after writeout**: Der Trace wird nach dem Sichern der Tracedaten wieder gestartet.

Mit **Trigger Delay** kann die Zeitdauer, während der der Trace nach Eintreffen einer Triggerbedingung noch aktiv ist, eingestellt werden.

Current Tasktrace Settings

Hier können Sie eine Einstellung sichern, laden oder löschen.

Speicherung der Traceeinstellungen

Die aktuellen Traceeinstellungen können in der XML-Datei "/user/simotion/hmi/files/persist/ttrace.xml" auf dem Speichermedium der Steuerung abgespeichert werden. Diese Datei wird im Hochlauf ausgewertet. Dadurch ist es möglich, auch das Tracen von Systemfunktionsaufrufen von der Weboberfläche aus zu aktivieren. Ergänzend bietet der Webserver die Möglichkeit, diese Datei zu löschen.

3.2.3.8 Diagnostic files

Diagnoseseiten des Webserver sichern

Die allgemeinen Diagnosedaten und einzelne HTML-Seiten von IT DIAG können über diese Seite gesichert werden.

Die Standard HTML-Seiten des Webserver enthalten wertvolle Informationen für die Analyse von Problemen, die bei dem Betrieb der SIMOTION Steuerung auftreten können.

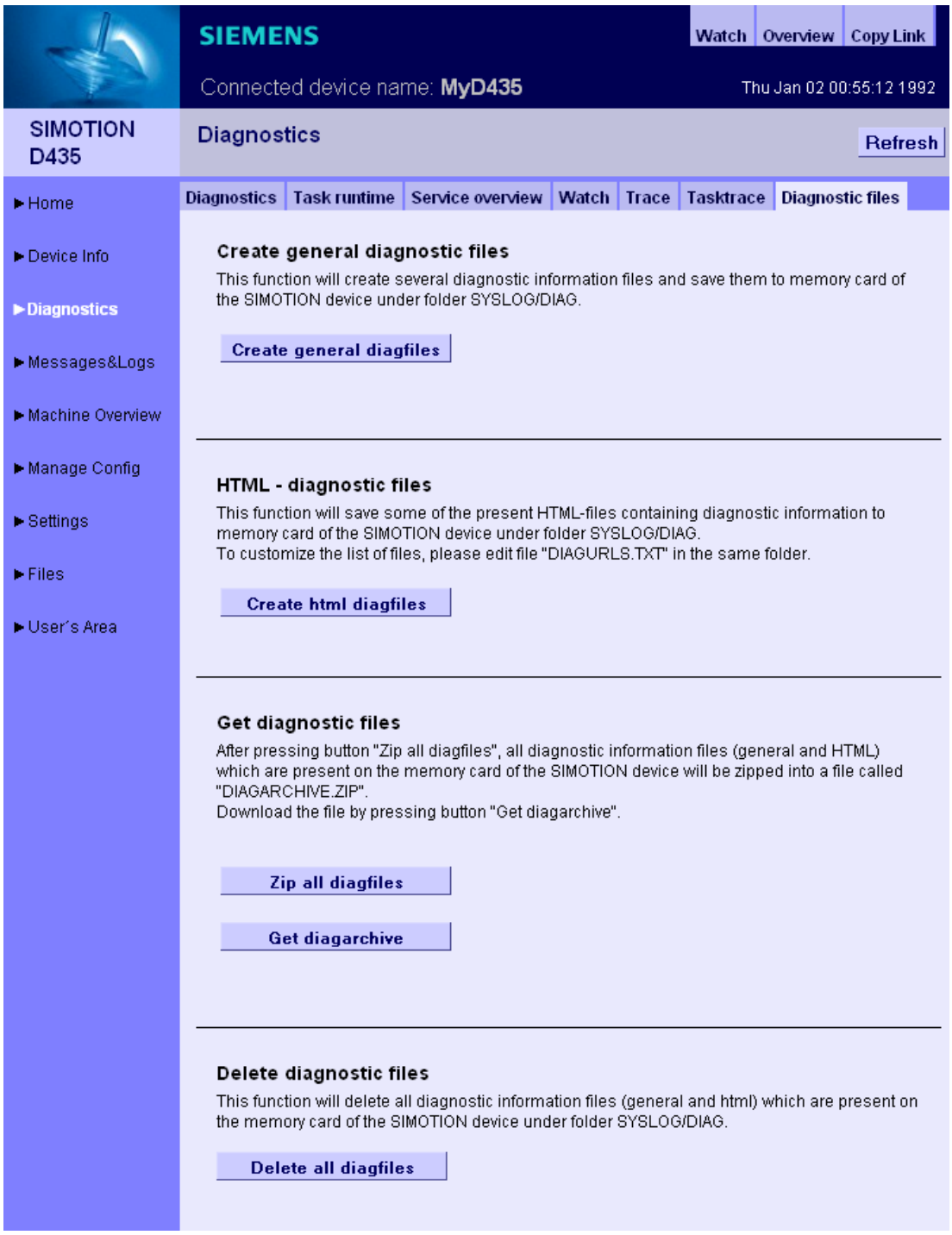


Bild 3-25 Diagnostic files

Create general diagnostic files
Diese Funktion speichert Diagnosedaten für den Support.

SIMOTION Gerät	Speichermedium	Pfad
D, C	CF Card/MMC	\USER\SIMOTION\HMI\SYSLOG\DIAG
P	Festplatte	F:\Simotion\user\Card\USER\SIMOTION\HMI\SYSLOG\DIAG
P320	CF Card	D:\Card\USER\SIMOTION\HMI\SYSLOG\DIAG

Die Benutzung dieser Funktion entspricht z. B. der Betätigung des Service-Wahlschalters bei der SIMOTION D Steuerung.
HTML-Dateien zur Diagnose werden nicht abgespeichert.

HTML - diagnostic files
Es wird eine Auswahl von relevanten Diagnoseseiten als HTML-Seiten auf dem Datenträger gesichert. Mit der Datei DIAGURLS.TXT (Seite 307) kann gesteuert werden, welche HTML-Seiten gesichert werden.

Zip all diagfiles
Diese Funktion zippt die Dateien, die zuvor durch das Anklicken der Buttons "Create general diagnostic files" und "HTML diagnostic files" erzeugt wurden. Die Zip-Datei ist leer, wenn vorher keine Dateien erzeugt wurden.

Get diagnostic files
Ermöglicht den Download der mit dem Button **Zip all diagfiles** erzeugten ZIP-Datei.

Delete all diagfiles
Löschung aller vorhandenen Diagnosedateien im Verzeichnis ...\USER\SIMOTION\HMI\SYSLOG\DIAG. Das Verzeichnis selbst bleibt bestehen.

3.2.4 Messages&Logs

3.2.4.1 Diag buffer

Diagnosepfuffer Informationen

Auf der Seite "Diag buffer" (zu erreichen über Messages&Logs > Diag buffer) erhalten Sie den aktuellen Inhalt des Diagnosepuffers der Steuerung.

Time	Zeitpunkt des Ereignisses
Date	Datum des Ereignisses
Event	Anzeige des Ereignisses als Text. Bei fehlender Sprachdatei DGBUFTXT.EDB erfolgt die Anzeige in Hexadezimaldarstellung

Hinweis

Voreingestellt sind englische Texte. Damit der Event-Text in einer anderen Sprache angezeigt wird, müssen die Dateien DGBUFTXT-XX.EDB und DGEXTXT.EDB in der jeweiligen Sprachversion auf die Speicherkarte der SIMOTION Steuerung in das Verzeichnis .../USER/SIMOTION/HMICFG übertragen werden. Siehe Gruppe DiagBuffer (Seite 254)

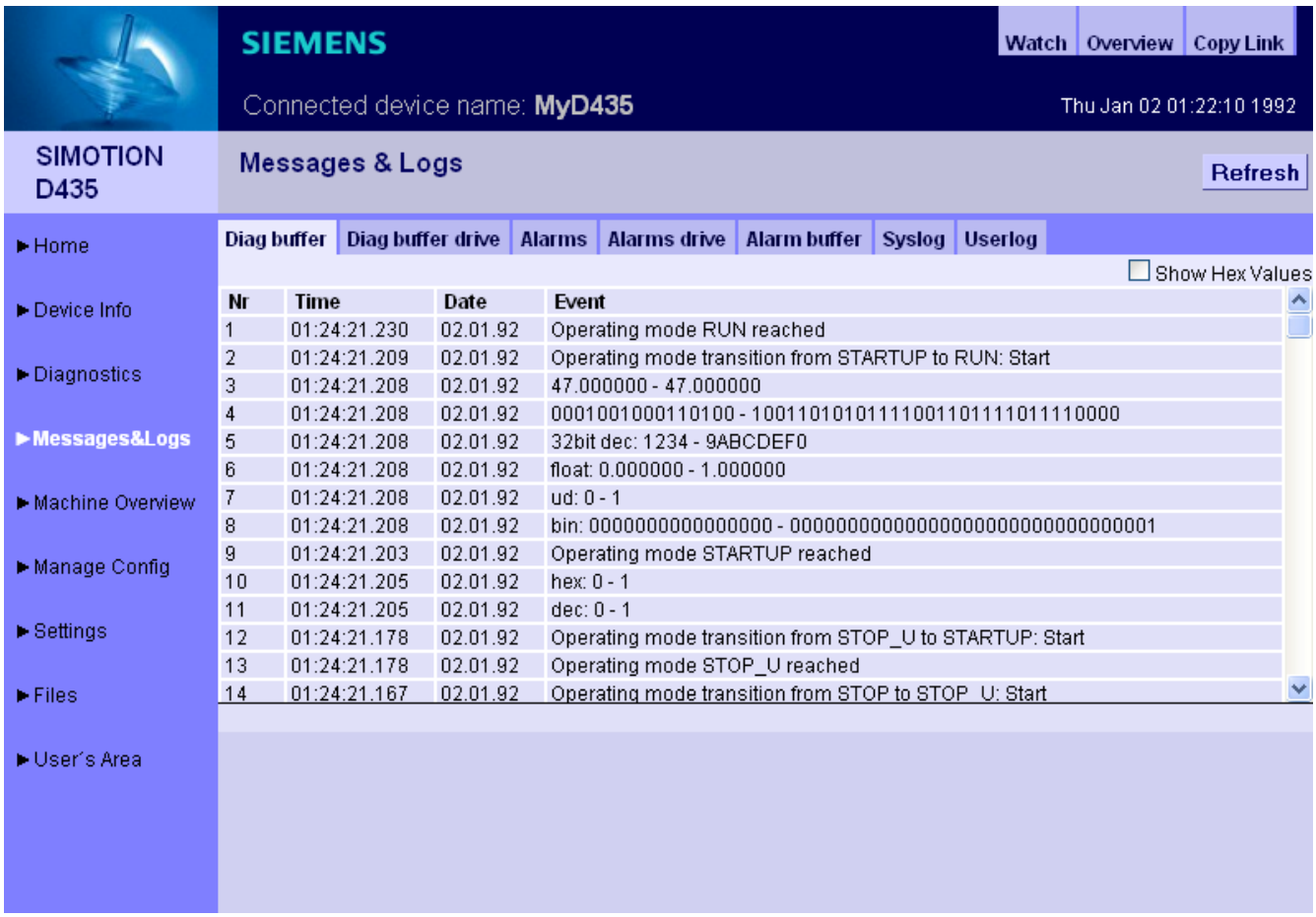


Bild 3-26 Diag buffer

3.2.4.2 Diag buffer drive

Darstellung des Antriebs-Diagnosepuffers

Analog zum SIMOTION Diagnosepuffer gibt es auch einen Diagnosepuffer für die integrierten Antriebe.

Time	Zeitpunkt des Ereignisses
Date	Datum des Ereignisses
Event	Anzeige des Ereignisses als Text.

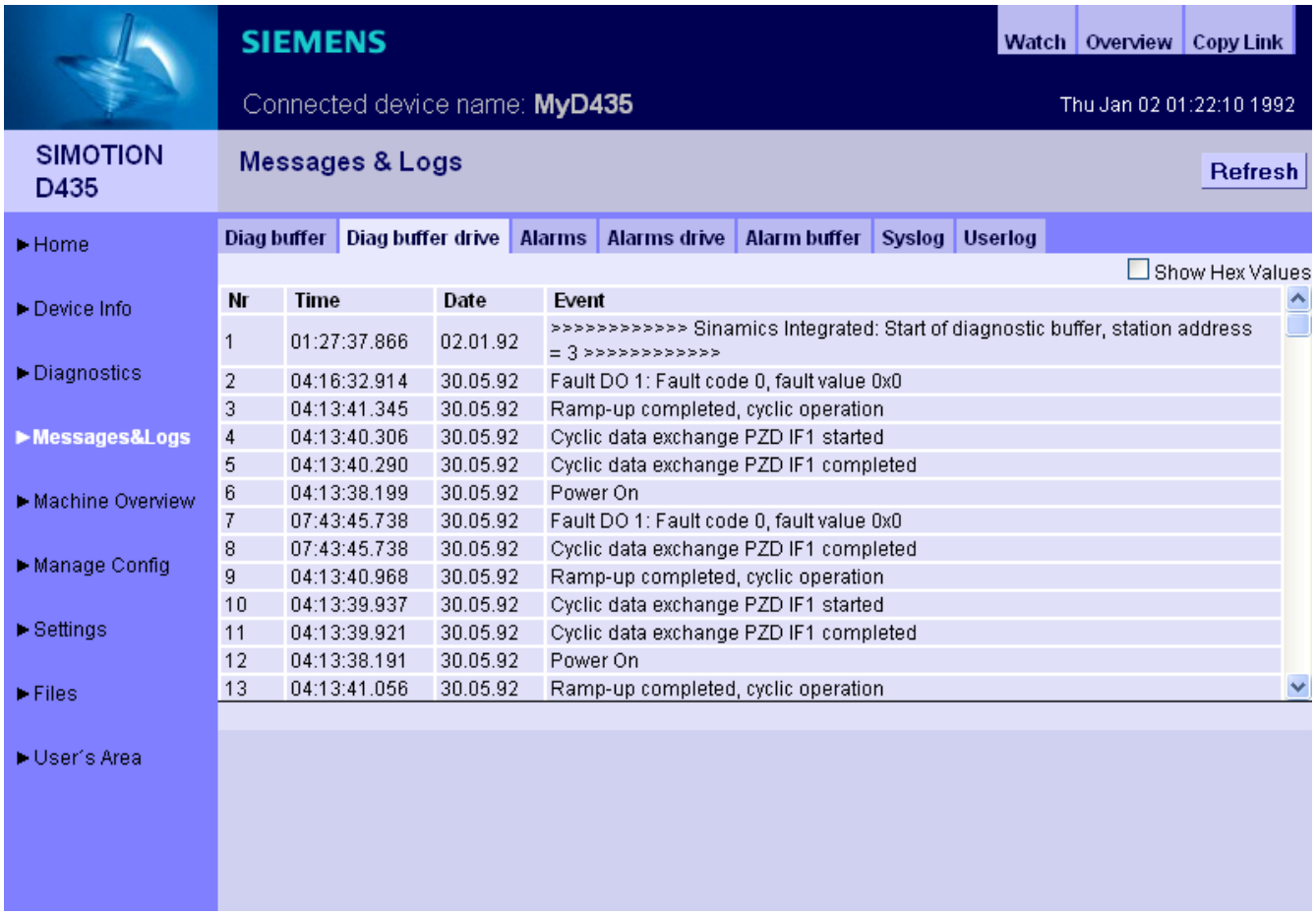


Bild 3-27 Anzeige des Diagnosepuffers der integrierten Antriebe

Der Diagnosepuffer einer Controller Extension CX32/CX32-2 kann auch auf diesem Weg angeschaut werden.

3.2.4.3 Alarms

Informationen zu Alarmen

Auf der Seite **Alarms** werden die Alarm- und AlarmS/SQ-Meldungen des Geräts angezeigt.

Tabelle 3- 1 Technological Alarms

Level	Kategorie des Alarms
Time	Zeitpunkt des Alarms
TO	Technologieobjekt, das den Alarm ausgelöst hat
Nr	Alarmnummer
Text	Anzeige der Alarmmeldung als Text

Tabelle 3- 2 Process Alarms (AlarmS/SQ)

AlarmNo	Nummer des AlarmS/SQ
State	Status des AlarmS/SQ
Time	Zeitpunkt an dem der AlarmS/SQ auftrat
Type	Typ des AlarmS/SQ
Text	Anzeige der Alarmmeldung als Text
More Info	Zusatzinformationen

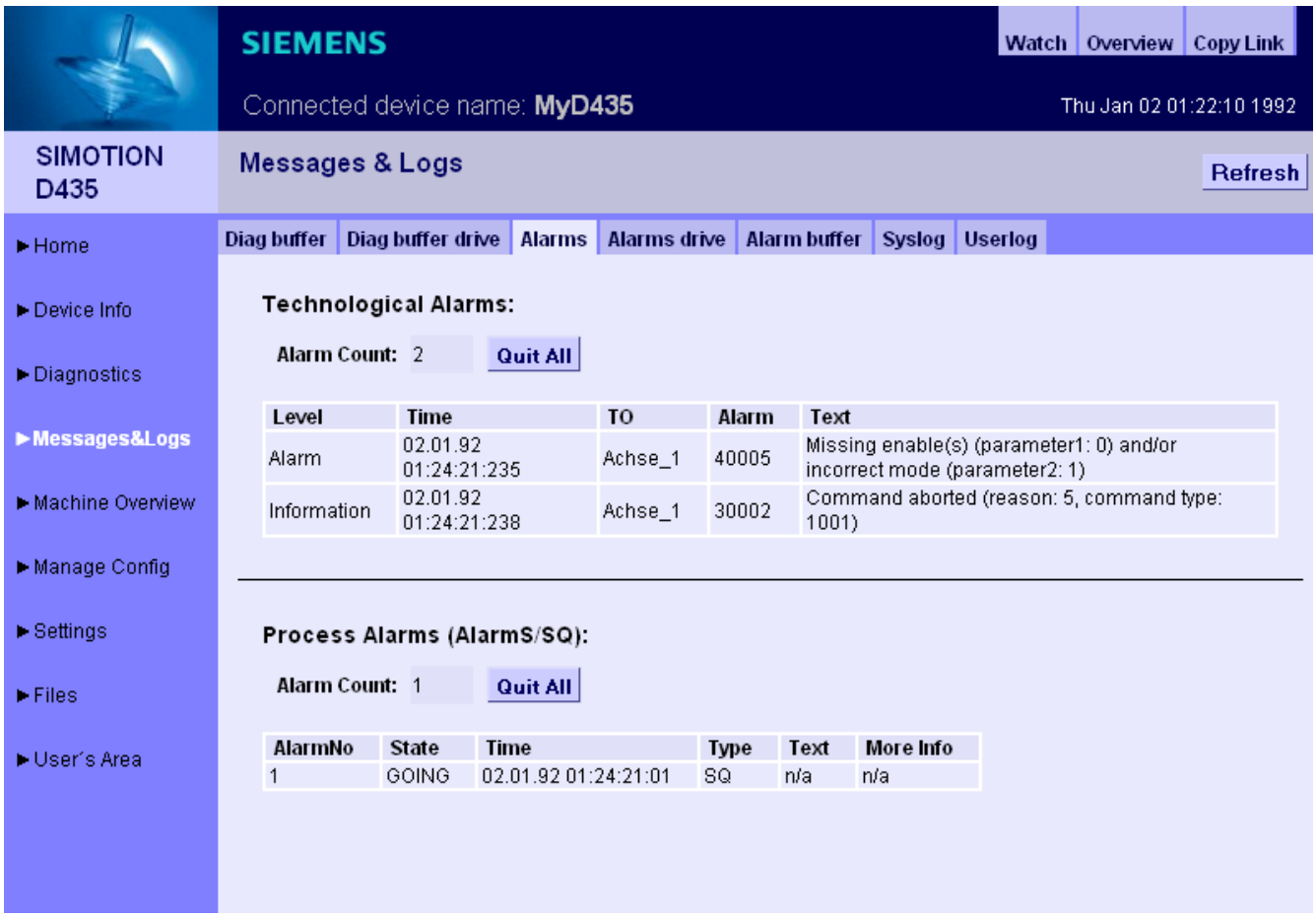


Bild 3-28 Alarms

Der Button **Quit All** ermöglicht es, alle quittierungspflichtigen Alarme zu schließen.

Spracheinstellung der Alarmtexte

Voreingestellt ist die Anzeige der englischen Alarmtexte. Damit die Alarmtexte in einer anderen Sprache angezeigt werden, muss die Datei TOALARM.ADB der entsprechenden Sprache auf die Speicherkarte der SIMOTION Steuerung übertragen werden.

Es kann immer nur eine Sprache auf der SIMOTION gespeichert werden.

Vorgehensweise

1. Öffnen Sie das Verzeichnis \AddOn\4_Accessories\Simotion_IT\4_Alarm_Messages\V4.2\ auf der DVD SIMOTION SCOUT Add-Ons. Bei der Sprache können Sie zwischen ger (Deutsch) und eng (Englisch), ita (Italienisch), fra (Französisch) wählen. In dem entsprechenden Verzeichnis finden Sie die TOALARM.ADB Datei.
2. Legen Sie die Speicherkarte der SIMOTION in ein Schreib-/Lesegerät ein.
3. Kopieren Sie die Datei TOALARM.ADB in das Verzeichnis \USER\SIMOTION\HMICFG. Wenn dieses Verzeichnis nicht existiert, erstellen Sie es.
4. Fügen Sie die Speicherkarte wieder in das SIMOTION Gerät ein.

Vorgehensweise P350

1. Beenden Sie die SIMOTION P.
2. Öffnen Sie das Verzeichnis \AddOn\4_Accessories\Simotion_IT\4_Alarm_Messages\V4.2\ auf der DVD SIMOTION SCOUT Add-Ons. Bei der Sprache können Sie zwischen ger (Deutsch) und eng (Englisch), ita (Italienisch), fra (Französisch) wählen. In dem entsprechenden Verzeichnis finden Sie die TOALARM.ADB Datei.
3. Kopieren Sie die Datei TOALARM.ADB in das Verzeichnis F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG (bei Default-Installation).
4. Starten Sie die SIMOTION P.

3.2.4.4 Alarms drive

Antriebsstörungen und Warnungen

Analog zu den technologischen Alarmen der Steuerung wird zusätzlich eine weitere Seite mit den Stör- und Warnmeldungen der Antriebe angeboten. Da Alarmtexte für Antriebsalarme derzeit nicht zur Verfügung stehen, erfolgt die Darstellung zunächst nur numerisch.

Dargestellt werden:

Time	Störzeit
Type	Fehlertyp
Source	DO-Name
No.	Störcode
Value	Störwert

Falls DOs (Drive Objects = Antriebsobjekte) namentlich im Antrieb vorhanden sind, werden sie auch namentlich ausgegeben.

Die Darstellung erfolgt in HEX (Es werden keine Alarmtexte ausgegeben).

Bild 3-29 DriveAlarms

Die Antriebsalarme der Controller Extension CX32/CX32-2 können ebenfalls angezeigt werden.

3.2.4.5 Alarm buffer

Inhalt des Alarmpuffers

Auf der Seite **Alarm buffer** erhalten Sie folgende Informationen:

Index	Nummerierung des Eintrags
Time	Zeitpunkt des Alarms
TO	Instanz des Technologieobjekts
Alarm	Alarmnummer
Text	

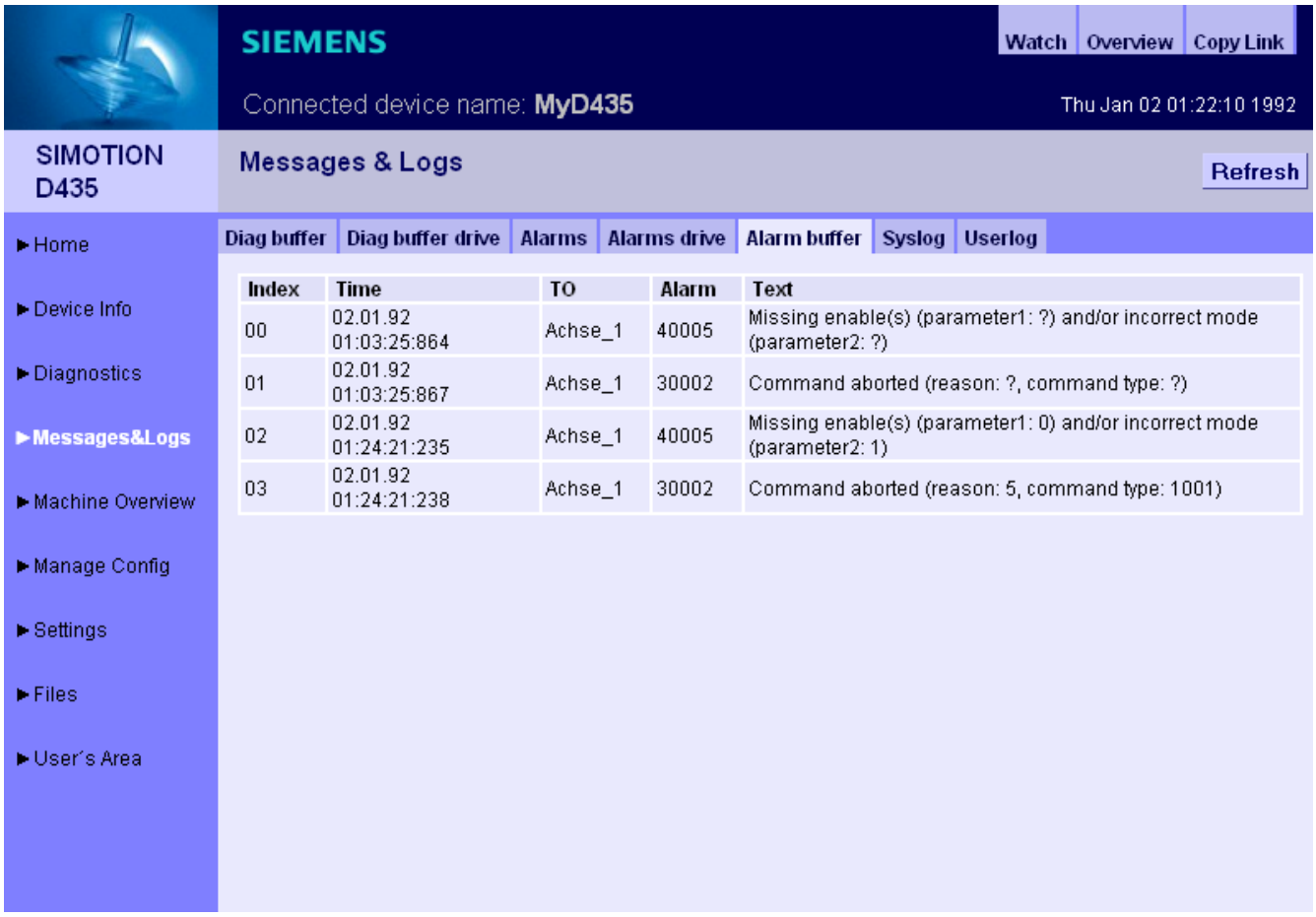


Bild 3-30 Anzeige des Alarmpuffers

Im Unterschied zur Seite **Alarms**, die die aktuell anstehenden Alarme anzeigt, wird auf der Seite **Alarm buffer** eine Historie aller Alarme dargestellt.

3.2.4.6 Syslog

Syslog

Auf der **Syslog** Seite wird die Syslog-Datei des betreffenden Geräts dargestellt.

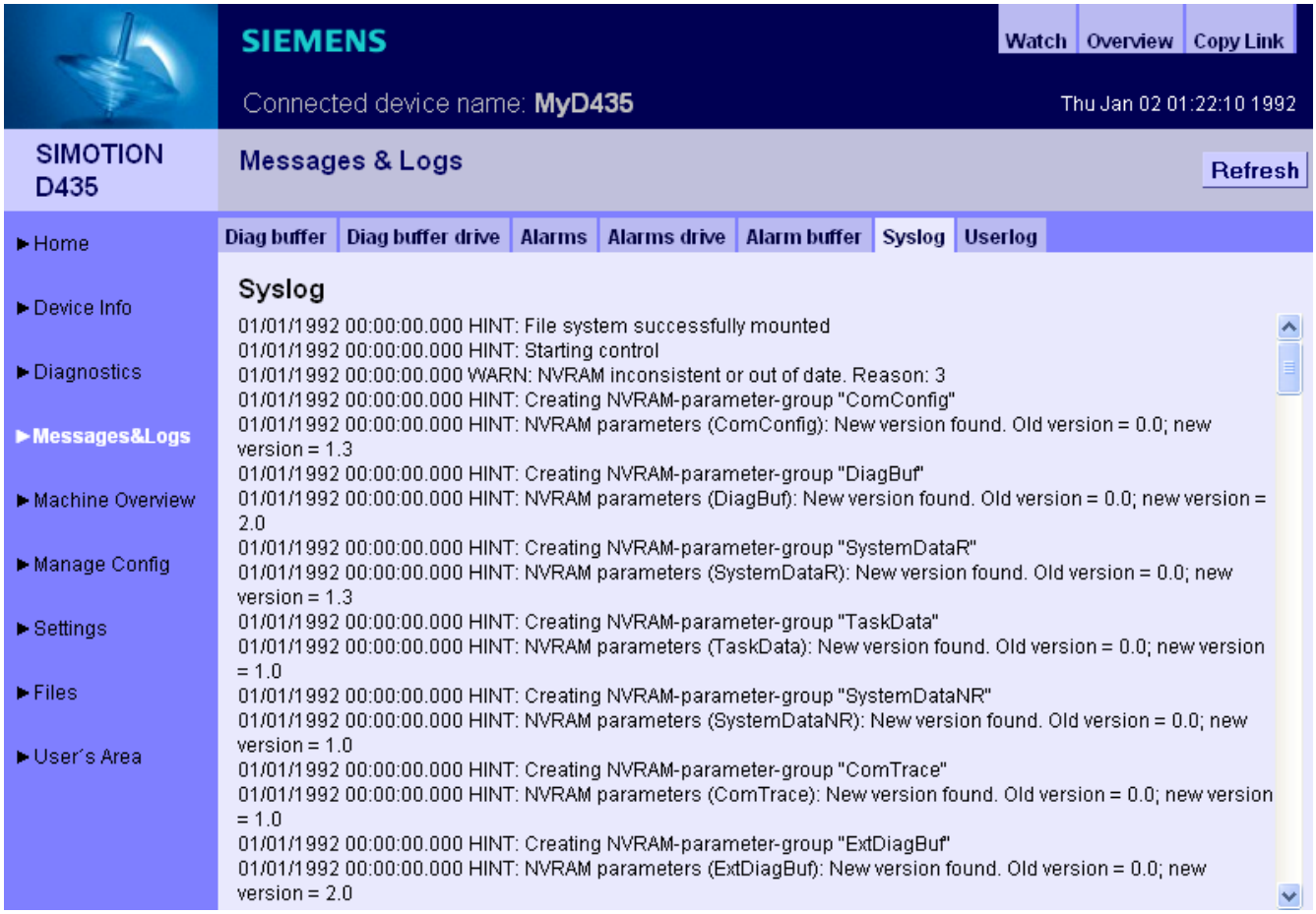


Bild 3-31 Syslog

Diese Datei wird vom System gepflegt. Es werden die für Diagnosezwecke wesentlichen Ereignisse dokumentiert, wie z. B. RAM2ROM.

3.2.4.7 Userlog

Userlog

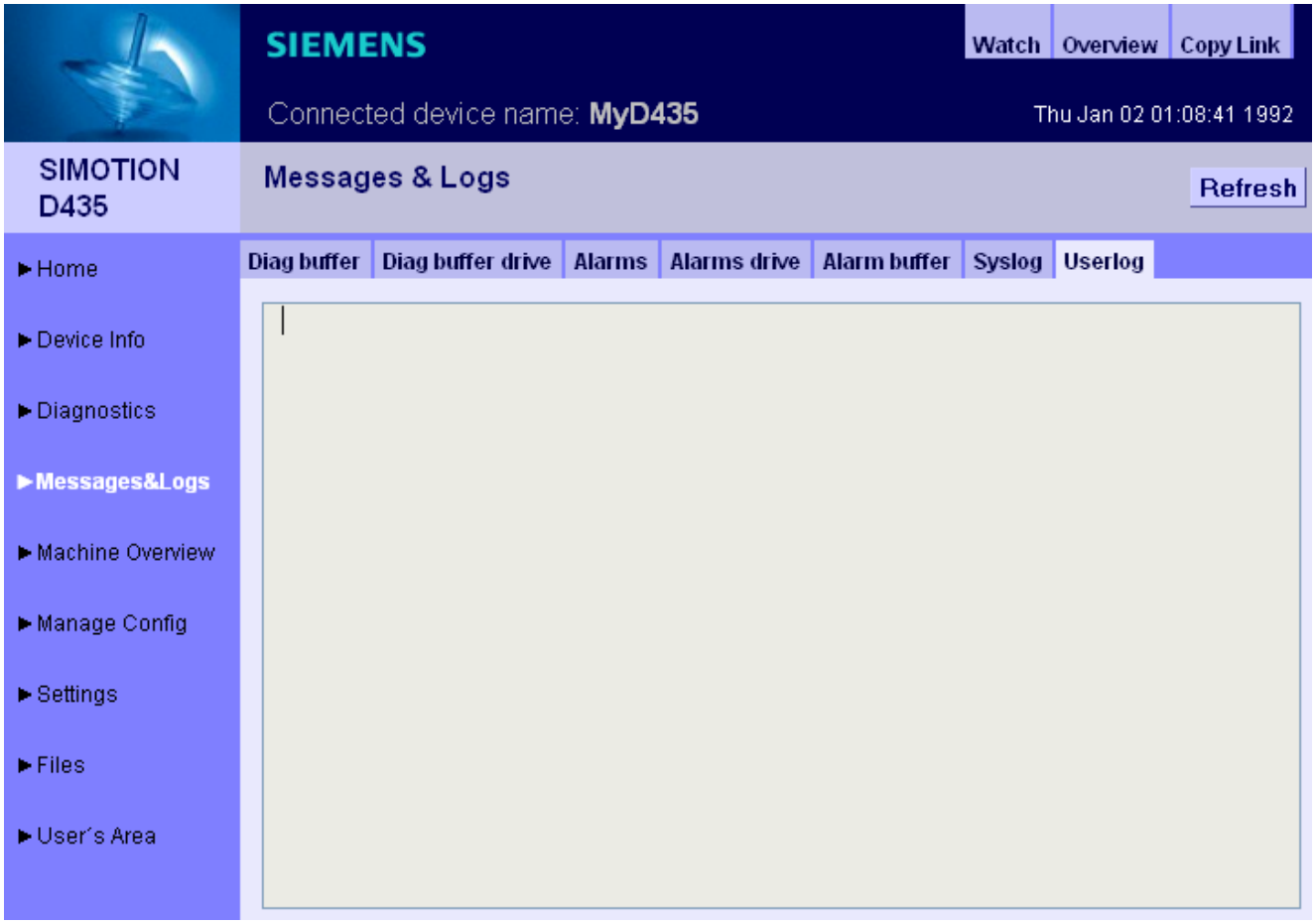


Bild 3-32 Userlog

Userlog zeigt Freitexte an, die vom Nutzer im SIMOTION SCOUT eingegeben wurden (**Gerätediagnose > Userlog**). Die Texte werden in einer Datei auf dem Speichermedium der Steuerung abgespeichert und auf der Webseite (schreibgeschützt) angezeigt.

3.2.5 Machine Overview

3.2.5.1 Module Information

Überblick über die projektierten Module

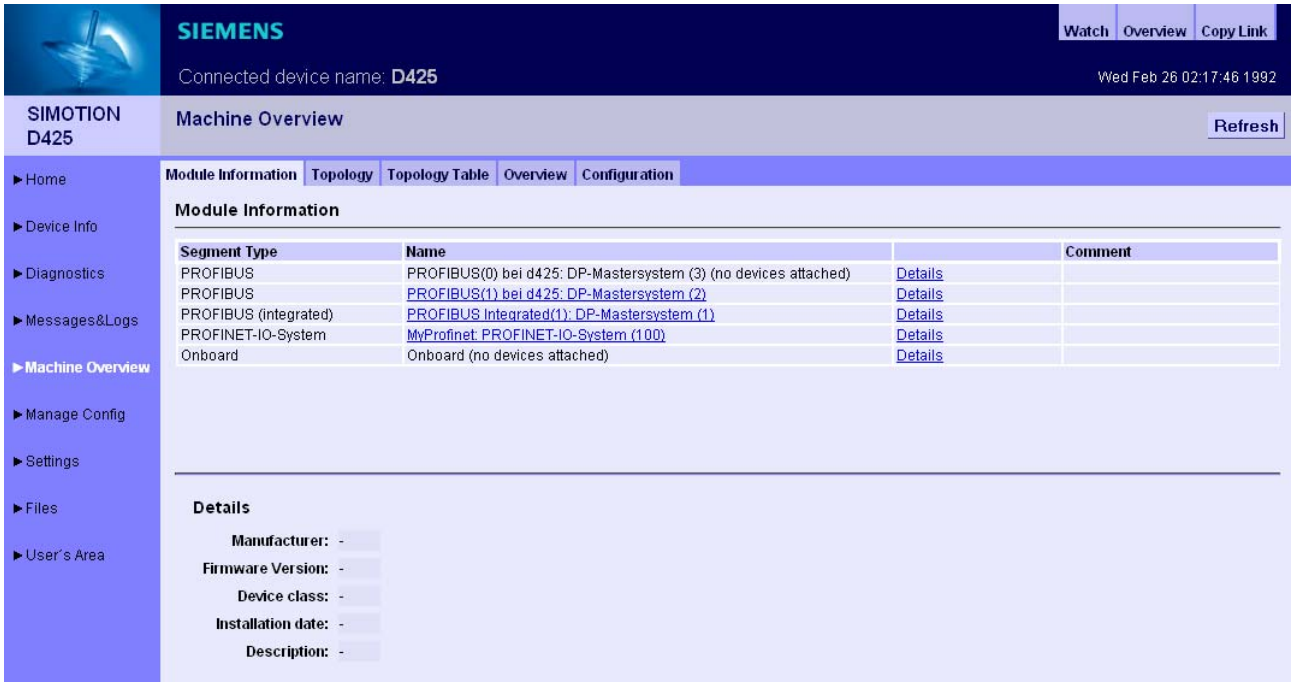


Bild 3-33 Module Information

Überblick über alle an der Maschine projektierten Module. Ausgehend vom Segment kann hierarchisch zum Element heruntergeklickt werden und Informationen dazu abgerufen werden.

Hinweis

Zur richtigen Darstellung der Informationen der **Machine Overview** Seiten ist es notwendig, dass eine HW Konfig in IT DIAG geladen wurde. Diese muss zum geladenen SCOUT Projekt passen, da sonst fehlerhafte Informationen angezeigt werden. Siehe Configuration (Seite 69)

Bild 3-34 Module Information Detailinformationen

Die Hierarchie ist immer: Segment > Device > Slot > Subslot (wenn vorhanden). Elemente, die keine Unterelemente haben, sind nicht anklickbar.

Klick auf das Segment zeigt alle Devices im Segment an (real shot).

Klick auf **Details** zeigt im unteren Bereich weitere Infos an (info2 shot).

Klick auf **back** geht eine Stufe zurück.

3.2.5.2 Topology

Übersicht der projektierten Topologie

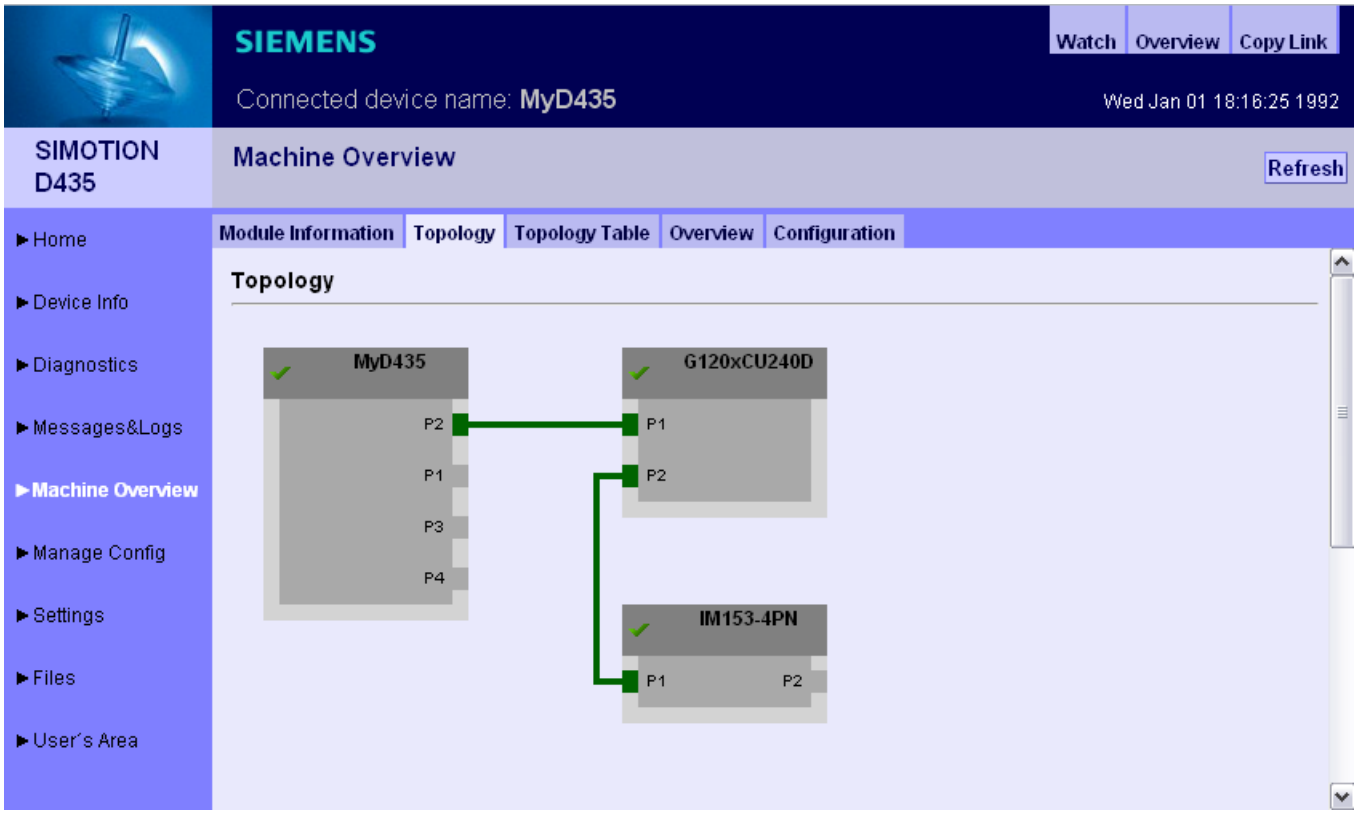


Bild 3-35 Topologie des Geräts

Auf dieser Seite wird die projektierte Topologie eines Geräts abgebildet. Nicht erreichbare Teilnehmer werden rot hinterlegt dargestellt.

Die Darstellung der Topologie zeigt, wie die Verkabelung der Teilnehmer aussehen muss.

3.2.5.3 Topology Table

Tabellarischer Übersicht der projektierten Topologie

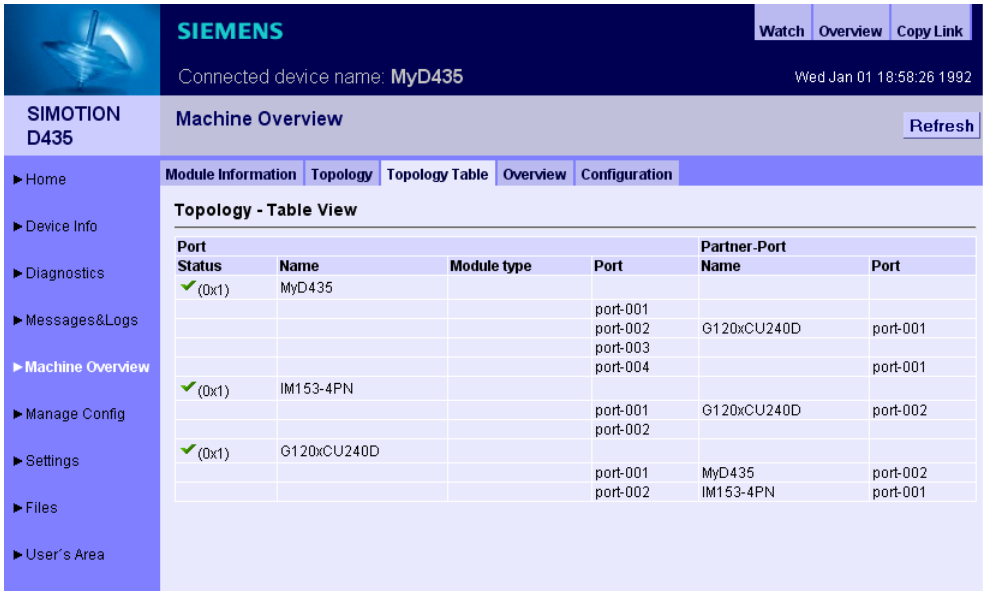


Bild 3-36 Tabellarische Topologie Tabelle

Diese Seite bietet einen schnellen Überblick über die Verkabelung in Textform.

Die angezeigten Informationen entsprechen denen der Topology (Seite 66) Seite.

3.2.5.4 Overview

Übersicht aller im Netz projektierten Module

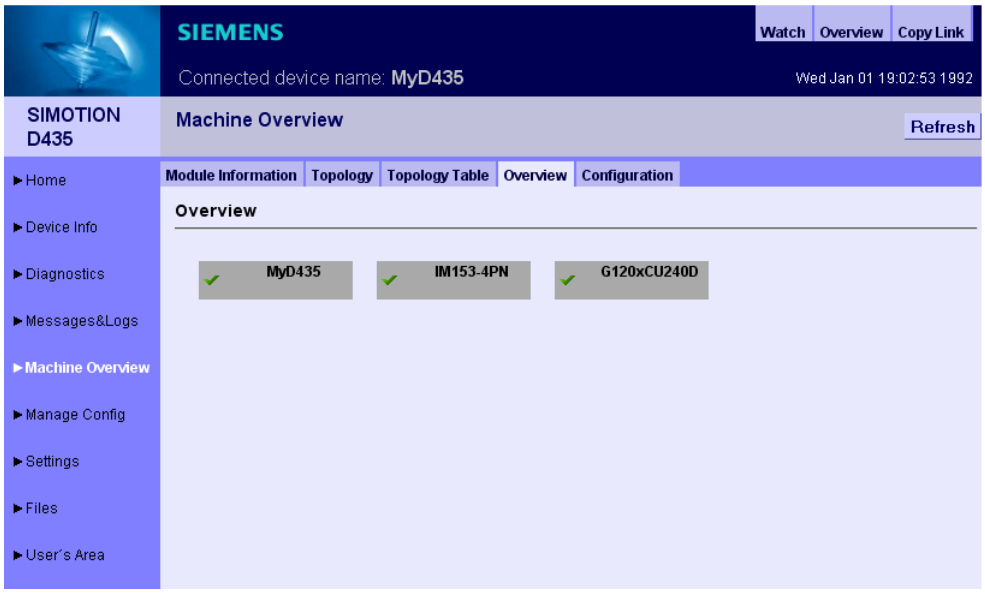


Bild 3-37 Overview

In dieser Übersicht werden alle im Netz projektierten Module ohne Topologieinformation angezeigt. Diese Übersicht ist vor allem für sehr große Projekte.

Nicht erreichbare oder ausgefallene Teilnehmer werden rot dargestellt.

3.2.5.5 Configuration

HW Konfig Information in IT DIAG laden

Eine HW Konfig Exportdatei muss in IT DIAG geladen werden. Erst dann sind alle Texte und Bezeichnungen der verbauten Module vorhanden. Die HW Konfig Exportdatei und das geladene SCOUT Projekt müssen zueinanderpassen, da sonst fehlerhafte Informationen angezeigt werden.

Export in HW Konfig

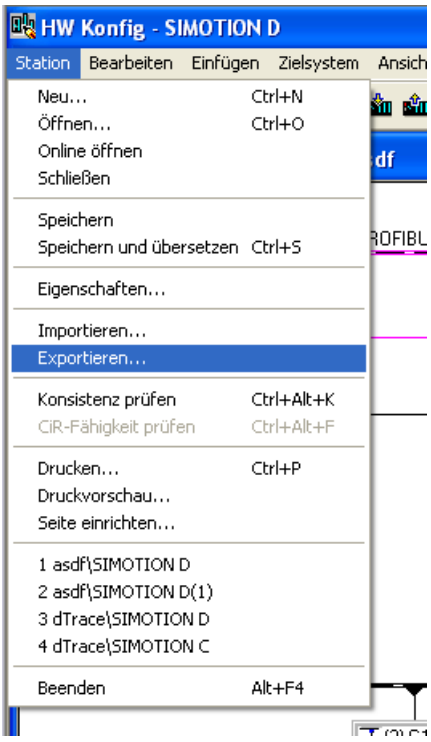


Bild 3-38 HW Konfig Export

- HW Konfig öffnen
- Menü **Station Exportieren**
- Speichern der Datei.
- Die Steuerung muss sich im Betriebszustand STOP befinden.
- Die entstandene Datei über das Formular der IT DIAG Seite laden.
- Danach findet ein Neustart der SIMOTION Steuerung statt.

Die Datei liegt dann auf der Karte im Verzeichnis /USER/SIMOTION/HMICFG/HWCONFIG.CFG.

Alternativ kann die Datei auch direkt auf die Karte mit einem Kartenlesegerät kopiert werden.

3.2.6 Manage config

3.2.6.1 Device Update

Device Update des Geräts

Diese Seite ermöglicht es ein Geräte-Update einzuspielen, und ausgewählte Daten vom Gerät auf den PC zu speichern.

Wenn mehrere Update Archive nacheinander in die Steuerung geschrieben wurden, besteht die Möglichkeit eine vorherige Projektierung wieder herzustellen.

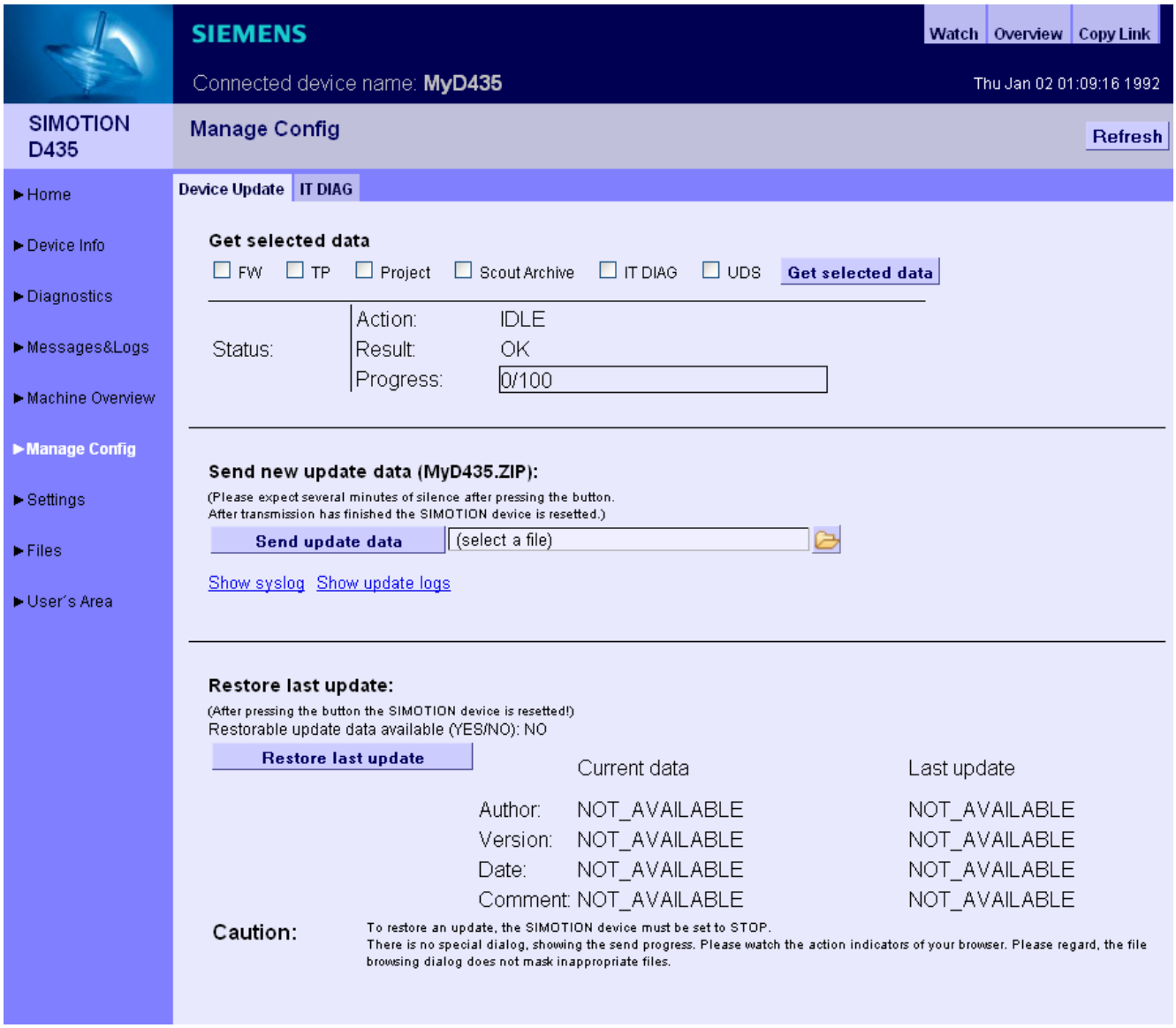


Bild 3-39 Manage Config

- **Get selected data** überträgt die aktuell aktiven Gerätedaten auf den PC. Die gesicherten Daten sind in einem Format, die die Rückspielung auf das Gerät ermöglichen.
  FW (Firmware), **TP** (Technologie Pakete), **Project** (aktuelles Projekt), **Scout Archive** (Inklusive der Scout Sicherung), **IT DIAG** (IT DIAG Konfiguration), **UDS** (Inklusive der **Unit Data Sets**)
- **Send new update data** überträgt eine mit dem Geräte Update Tool erzeugte Datei auf das Gerät. Dieser Vorgang kann mehrere Minuten dauern und führt zu einem Neustart des Geräts.
- **Restore last update** reaktiviert den letzten Stand der Gerätedaten des vorangegangenen Software Updates.

Weitere Informationen zu diesem Thema finden Sie in der Betriebsanleitung 'SIMOTION Geräte hochrüsten'.

Hinweis

Der Download der Firmware wird von der SIMOTION Steuerung P nicht unterstützt.

⚠ GEFAHR

Um ein Projekt oder eine Firmware zu senden oder herunterzuladen, muss die Steuerung in den STOP-Zustand gebracht werden.

Typ und Inhalt der Datei werden beim Übertragen nicht geprüft.

Bei einer ungültigen Konfiguration muss auf der Speicherkarte das USER-Verzeichnis gelöscht werden.

Hinweis

Bei kleinen Kärtchen (32 MB/64 MB) kann es beim Update zu Problemen kommen, wenn nicht genügend Speicherplatz zur Verfügung steht.

Der erforderliche Speicherplatzbedarf ergibt sich aus der Größe der bestehenden Konfiguration und des Updates zusammen.

Je nach betroffener Datei führt die SIMOTION Steuerung nach dem Betätigen der Schaltfläche "Send update data" die folgenden Aktionen automatisch durch:

- WebCfg.xml

  Neu Starten des Webservers.

  Hinweis: Alle OPC XML-DA Subscriptions gehen verloren.
- MyProject.ZIP

  Speichern des neuen Projekts samt Ethernet-Konfiguration in die (virtuelle) Speicherkarte und aktivieren des neuen Projekts durch einen Neustart der SIMOTION Steuerung.
- XXXXXXFW.ZIP

  Speichern der Firmware auf die Speicherkarte und aktivieren der neuen Firmware durch einen Neustart der SIMOTION Steuerung.

Der Zugriff auf diese Seite ist nur als angemeldeter Benutzer möglich. Siehe Loginverwaltung (Seite 18)

Verwendung älterer Konfigurationsdaten

Ältere Konfigurationsdaten, die mit der SIMOTION SCOUT Funktion **Laden ins Dateisystem** erstellt wurden, können weiterhin über IT DIAG eingespielt werden.

Das vom SCOUT hierbei erzeugte ZIP-File kann über **Send update data** auf das Gerät übertragen werden.

3.2.6.2 Hochrüsten der Firmware vor V4.2

Das Hochrüsten der Firmware kann bis zur Version 4.2 zu folgender Situation führen: Eine alte WebCfg.xml bleibt auf dem Gerät erhalten und führt zur Darstellung leerer Diagnoseseiten.

Möglichkeit dieses Problem zu vermeiden:

- Explizites Löschen der WebCfg.xml.

Nach dem nächsten Reset wird vom Gerät eine neue WebCfg.xml erzeugt. Die alte WebCfg.xml sollte vorher gesichert werden, um aus der alten Konfiguration Einstellungen in die neue WebCfg.xml übernehmen zu können.

Siehe auch

Device Update (Seite 70)

3.2.6.3 Editierfunktion

Editierfunktionen der IT DIAG Seiten

Auf einigen Standard Seiten kann die Konfigurationsdatei WebCfg.xml über den Browser editiert werden. Die Editierfunktionen sind immer gleich aufgebaut und werden in diesem Abschnitt erklärt.

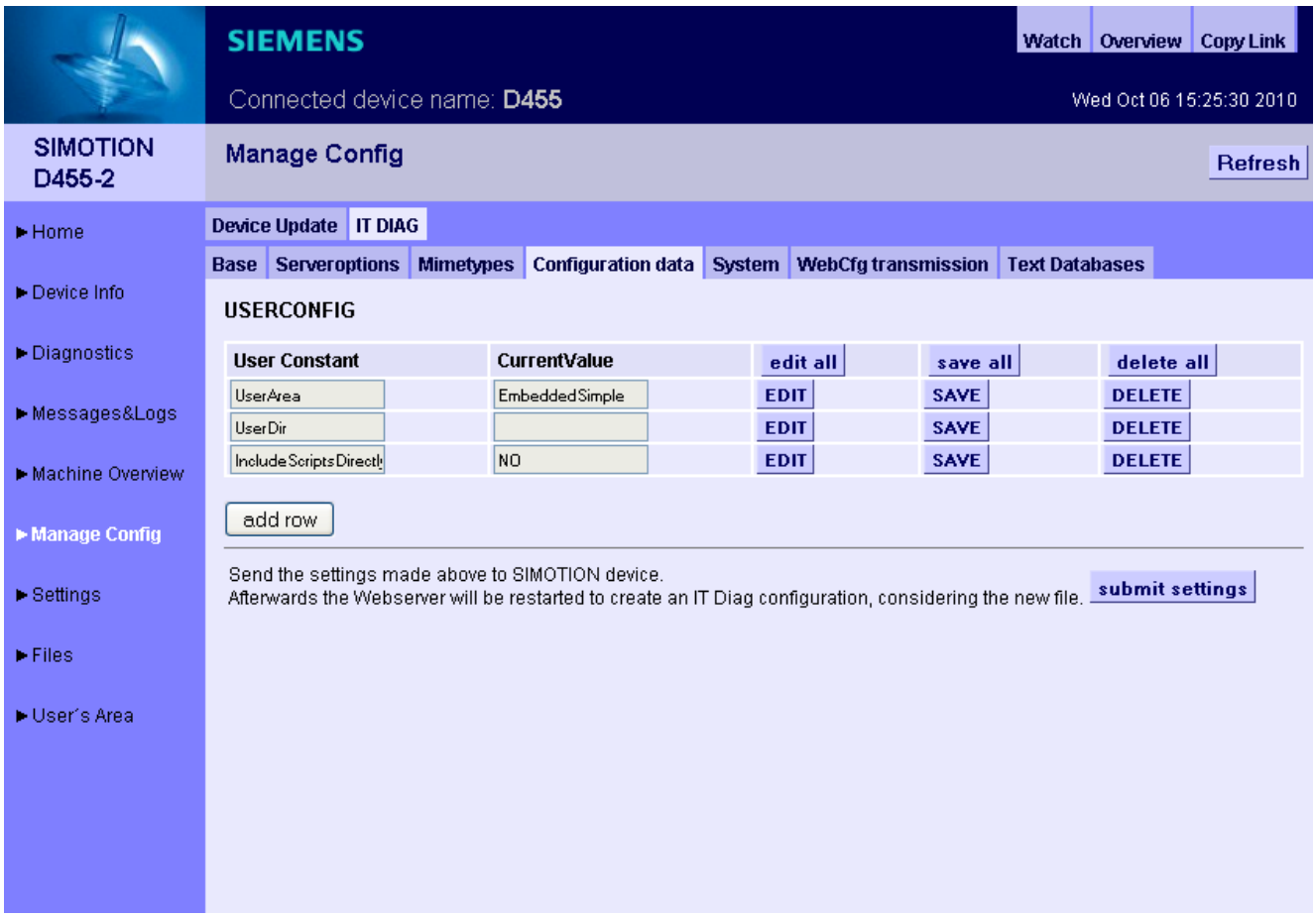


Bild 3-40 Editierfunktionen 1

Der Button **add row** fügt eine Zeile ein.

Um die Zeile zu ändern, muss zuerst der **EDIT** Button der betreffenden Zeile angeklickt werden. Die Eingabefelder können dann ausgefüllt werden.

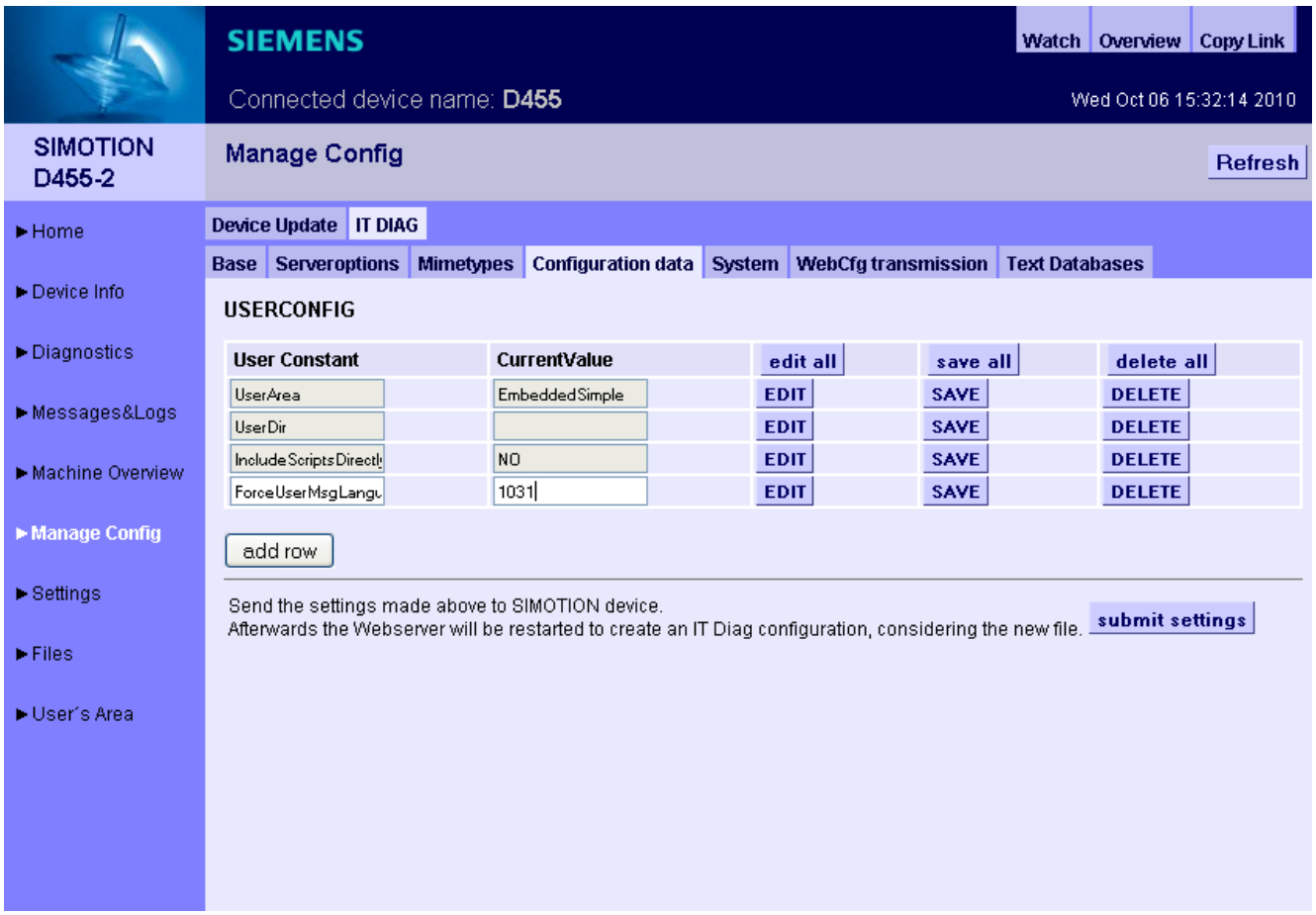


Bild 3-41 Editierfunktionen 2

Nach dem Editieren müssen die Änderungen mit dem **SAVE** Button in IT DIAG gespeichert werden. Mit dem **SAVE** Button werden die Änderungen nur im Browser gespeichert und noch nicht an das Gerät übertragen. Für die endgültige Übertragung zum Gerät muss der **submit settings** Button benutzt werden.

Der **DELETE** Button löscht die Eingaben in der betreffenden Zeile der Seite, aber nicht auf dem Gerät.

Mit **edit all**, **save all** und **delete all** können alle Angaben in der Seite editiert, gespeichert oder gelöscht werden.

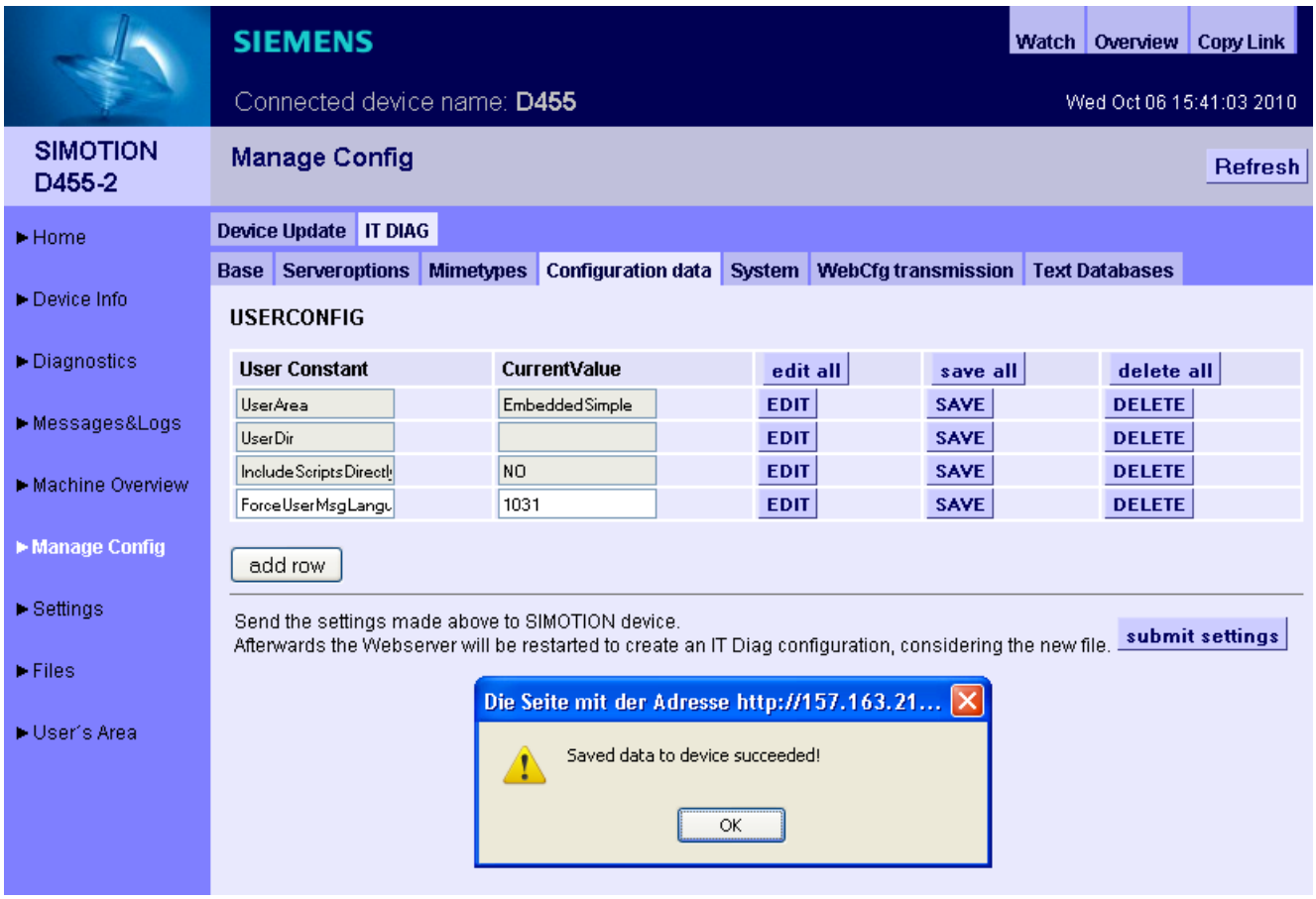


Bild 3-42 Editierfunktionen3

Zur Übertragung der geänderten Daten auf das Gerät muss der **submit settings** Button angeklickt werden. Dieser Vorgang wird mit einer entsprechenden Meldung bestätigt und der Webserver des Geräts mit der geänderten WebCfg.xml neu gestartet.

3.2.6.4 IT DIAG Reiter

Webseiten zur Änderung der Konfiguration

Der Reiter IT DIAG fasst die Webseiten zusammen, die zur Konfiguration des IT DIAG dienen.

Alle Einstellungen führen zu Änderungen in der Datei WebCfg.xml. Alternativ zum Editieren mithilfe der Webseiten, kann diese XML-Datei auch direkt angepasst werden.

3.2.6.5 IT DIAG Base

Einstellen der LOCALLINKS

Bild 3-43 IT DIAG Base

Der Reiter **Base** ermöglicht die Editierung der LOCALLINKS im `<BASE>`-Tag der WebCfg.xml.

Über LOCALLINKS (Seite 110) kann auf das physikalische Dateisystem zugegriffen werden.

Attribute	Typ	Beispiel
FILENAME	Zeichenkette	Index.mbs
REALM	Zeichenkette	Ein Gruppenname: Administrator
LOCALLINK	Zeichenkette	LOCALLINK="$WWWROOT/NewDir/index.mbs "
PREFER_EXTERNAL	TRUE/FALSE	
BROWSEABLE	TRUE/FALSE	
READ	Zeichenkette	Ein oder mehrere Gruppennamen: Administrator,Servicegroup
WRITE	Zeichenkette	Ein oder mehrere Gruppennamen: Administrator,Servicegroup
MODIFY	Zeichenkette	Ein oder mehrere Gruppennamen: Anyone

Attributübersicht Reiter Base

Siehe auch

Attribut LOCALLINK (Seite 282)

3.2.6.6 IT DIAG Serveroptions

Grundeinstellungen

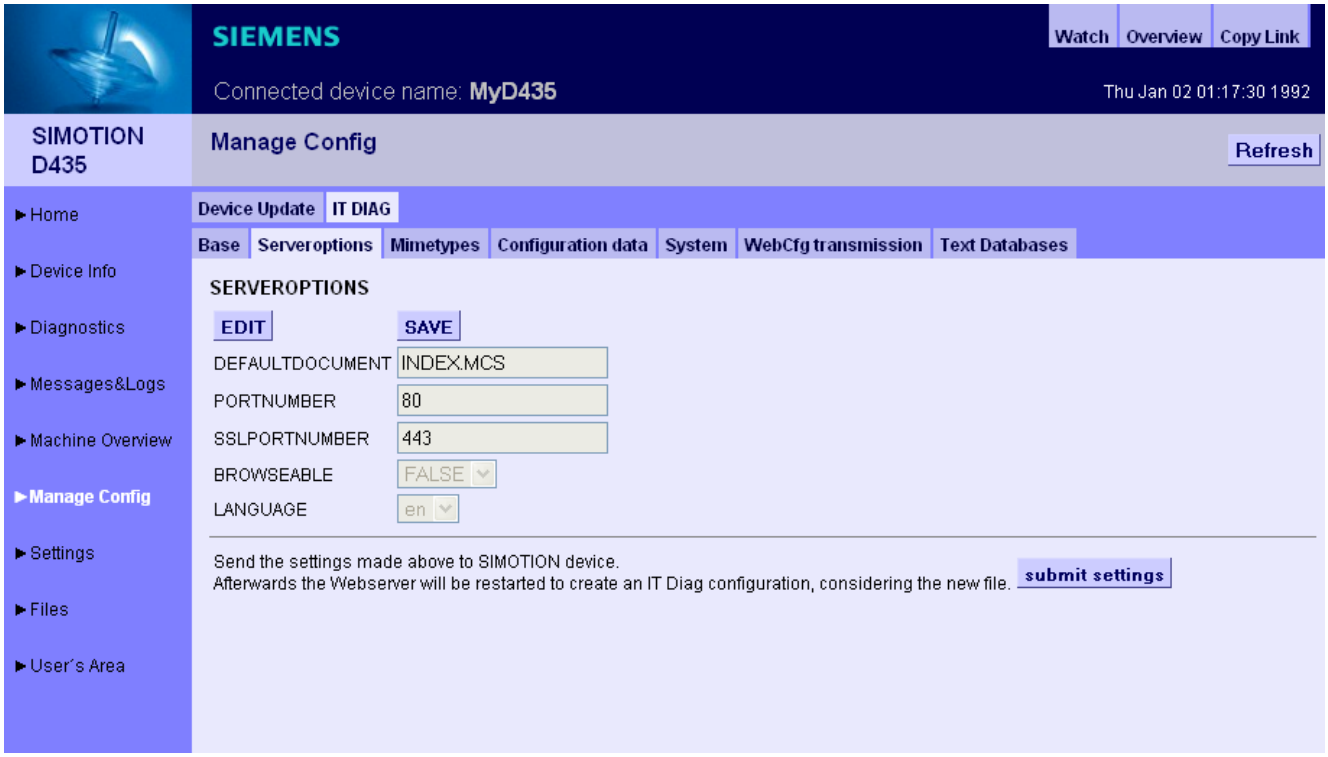


Bild 3-44 IT DIAG Serveroptions

Dieses Register ermöglicht die Einstellung grundlegender Parameter des Webservers.

Es werden verschiedene Einstellungen des `<SERVEROPTIONS>`-Tag in der WebCfg.xml mit dieser Seite gesetzt.

- DEFAULTDOCUMENT (Seite 277) ermöglicht die Änderung der Startseite. Voreingestellt ist INDEX.MCS.
- PORTNUMBER (Seite 279) legt den TCP/IP Port für die Ausgabe der Webserver-Seiten fest. Voreingestellt ist Port 80 (http).
- SSLPORTNUMBER (Seite 280) legt den TCP/IP Port für die verschlüsselte Ausgabe der Webserver-Seiten fest. Voreingestellt ist Port 443 (https).

Nicht änderbare Angaben

- BROWSEABLE (Seite 275) zeigt die Einstellung der Verzeichnisanzeige.
- LANGUAGE zeigt die Spracheinstellung.

3.2.6.7 IT DIAG Mimetypes

MIME-Typen

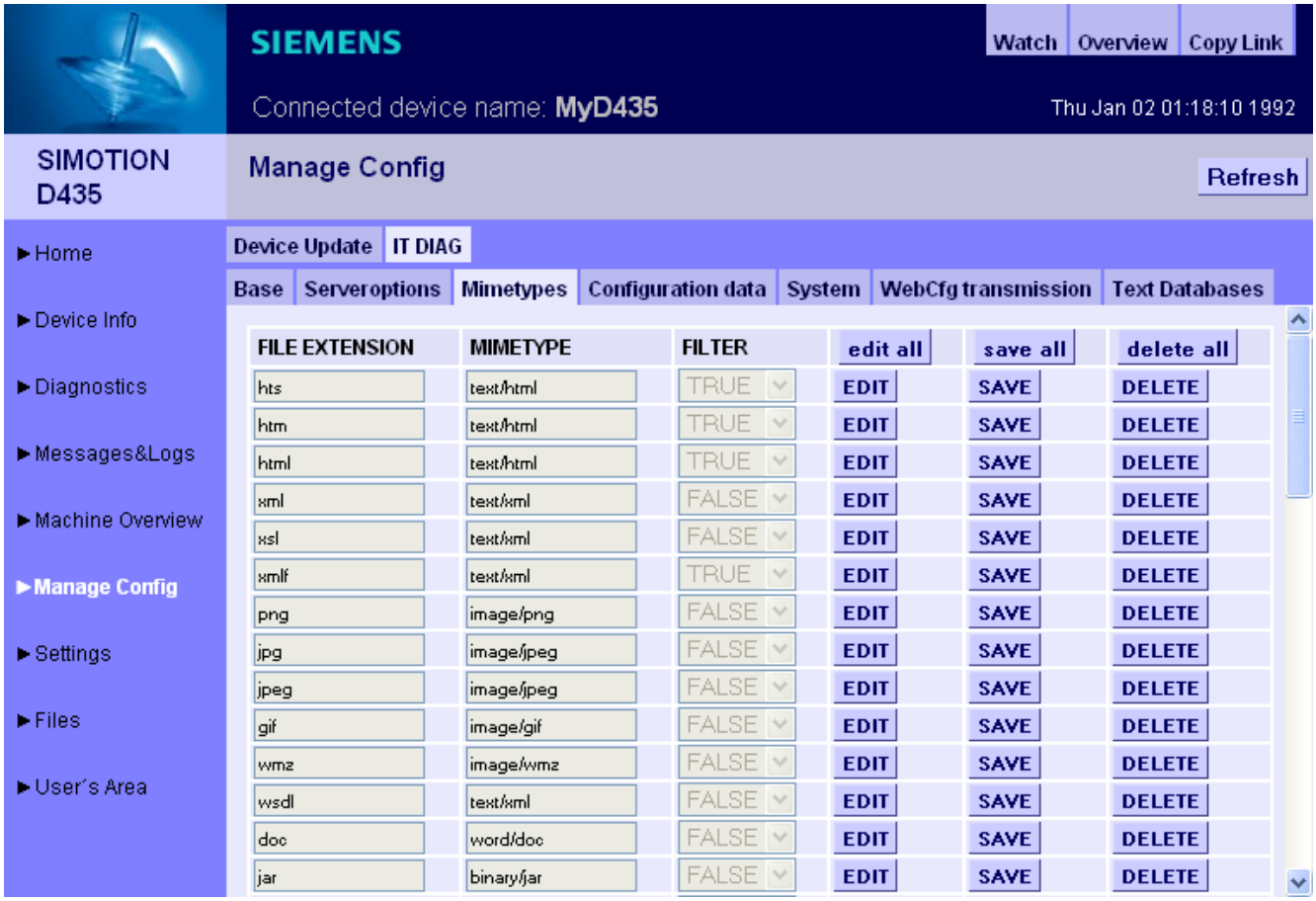


Bild 3-45 IT DIAG Mimetypes

In diesem Register kann ein MIME-Typ mit einer Dateiendung verbunden werden.

Über den MIME-Typ wird dem Browser im HTTP-Header signalisiert, welche Art Daten übertragen werden.

Durch die Einstellung in der Spalte Filter kann festgelegt werden, ob betreffende Dateien durch die Serverfilter MWSL und SSI bearbeitet werden und darin enthaltene Tags entsprechend ausgewertet bevor sie an den Browser gesendet werden.

Siehe auch

<MIME_TYPES> (Seite 278)

3.2.6.8 IT DIAG Configuration data

Konfiguration benutzerdefinierter Konstanten

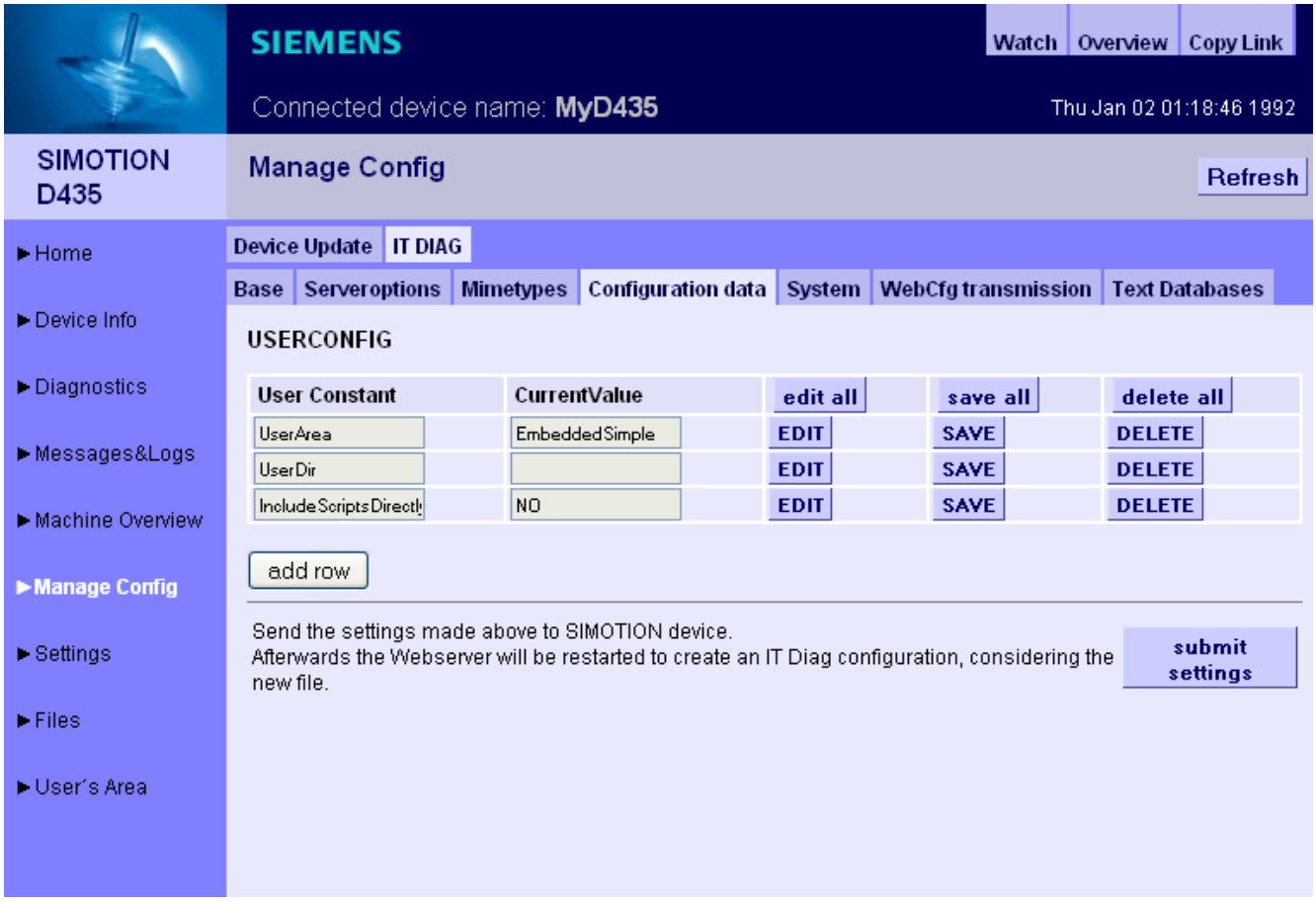


Bild 3-46 IT DIAG Configuration data

Diese Seite ermöglicht die Anlage und Editierung von Konfigurationskonstanten.

Siehe auch

Konfigurationskonstanten (Seite 170)

3.2.6.9 IT DIAG System

Benutzerdatenbank

Die System-Seite ermöglicht die Benutzerverwaltung. Es können Passwörter, Gruppen- und Zugriffsrechte für die Benutzer vergeben werden.

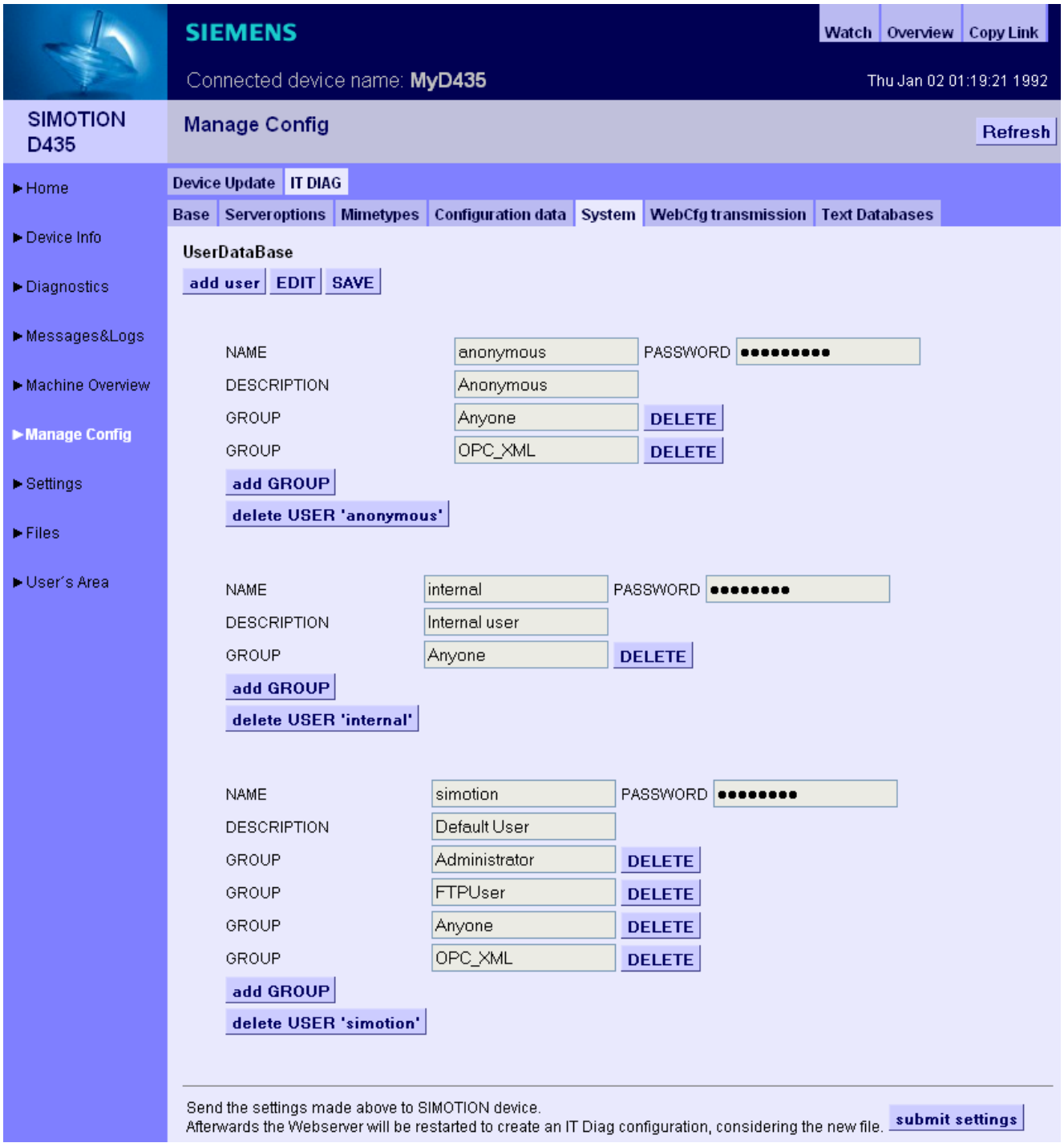


Bild 3-47 Benutzerdatenbank

Der Button **add user** legt einen neuen Benutzer an. **EDIT** ermöglicht das Editieren und **SAVE** die Speicherung der Benutzerangaben.

Mit dem Button **add GROUP** können einem Benutzer Gruppen zugeordnet werden. Der Button **delete USER '<Benutzername>'** löscht den entsprechenden Benutzer.

Siehe auch

Loginverwaltung (Seite 18)

<USERDATABASE> (Seite 281)

3.2.6.10 IT DIAG WebCfg Transmission

Übertragung der Konfigurationen an das Gerät

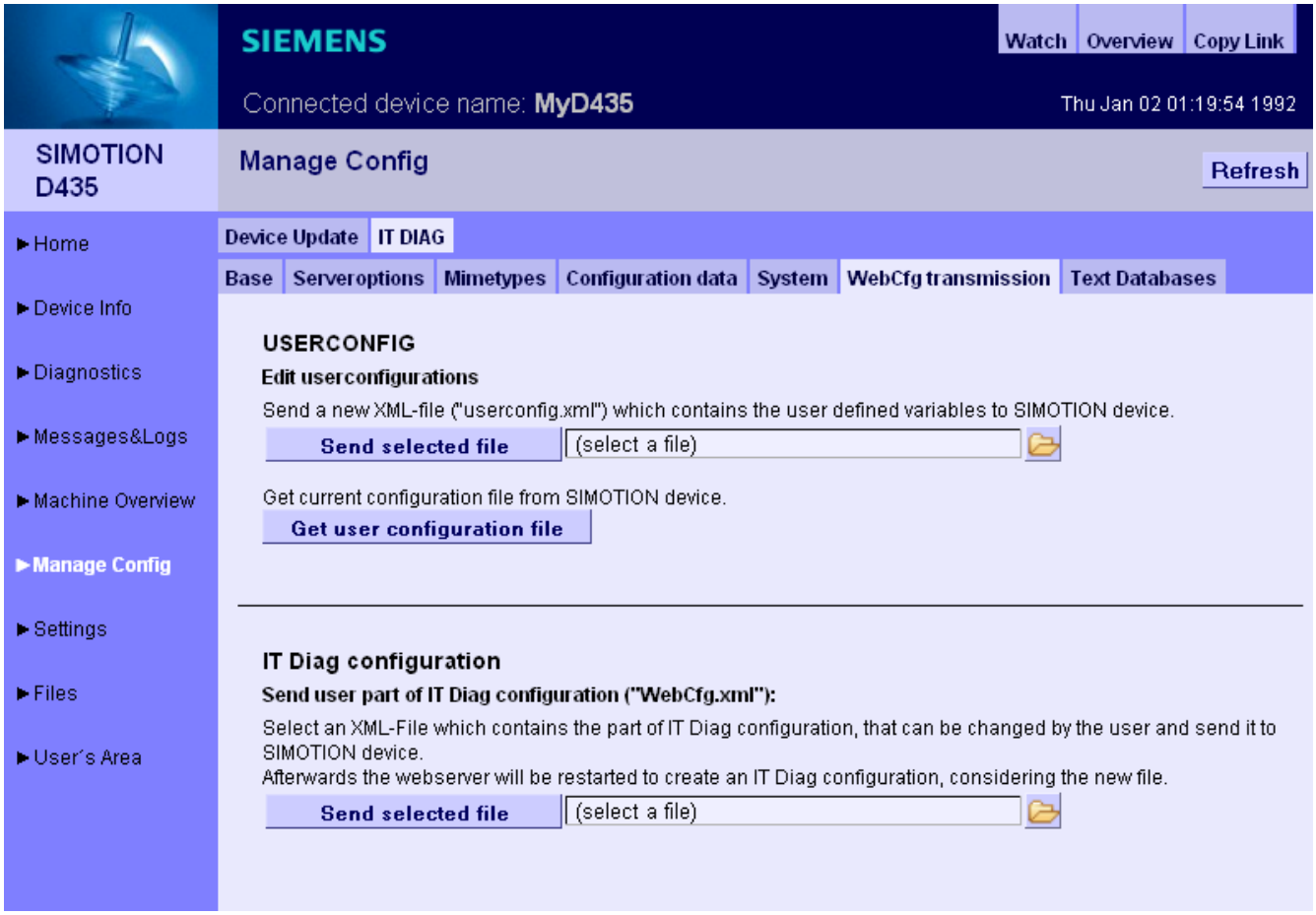


Bild 3-48 WebCfg transmission

Die Benutzerdaten können über diese Seite an das Gerät gesendet bzw. von ihm empfangen werden.

Außerdem ermöglicht diese Seite eine lokal bearbeitete WebCfg.xml auf das Gerät zu übertragen.

3.2.6.11 IT DIAG Text Databases

Übertragung benutzerdefinierter Meldungen vom SIMOTION SCOUT zum Gerät

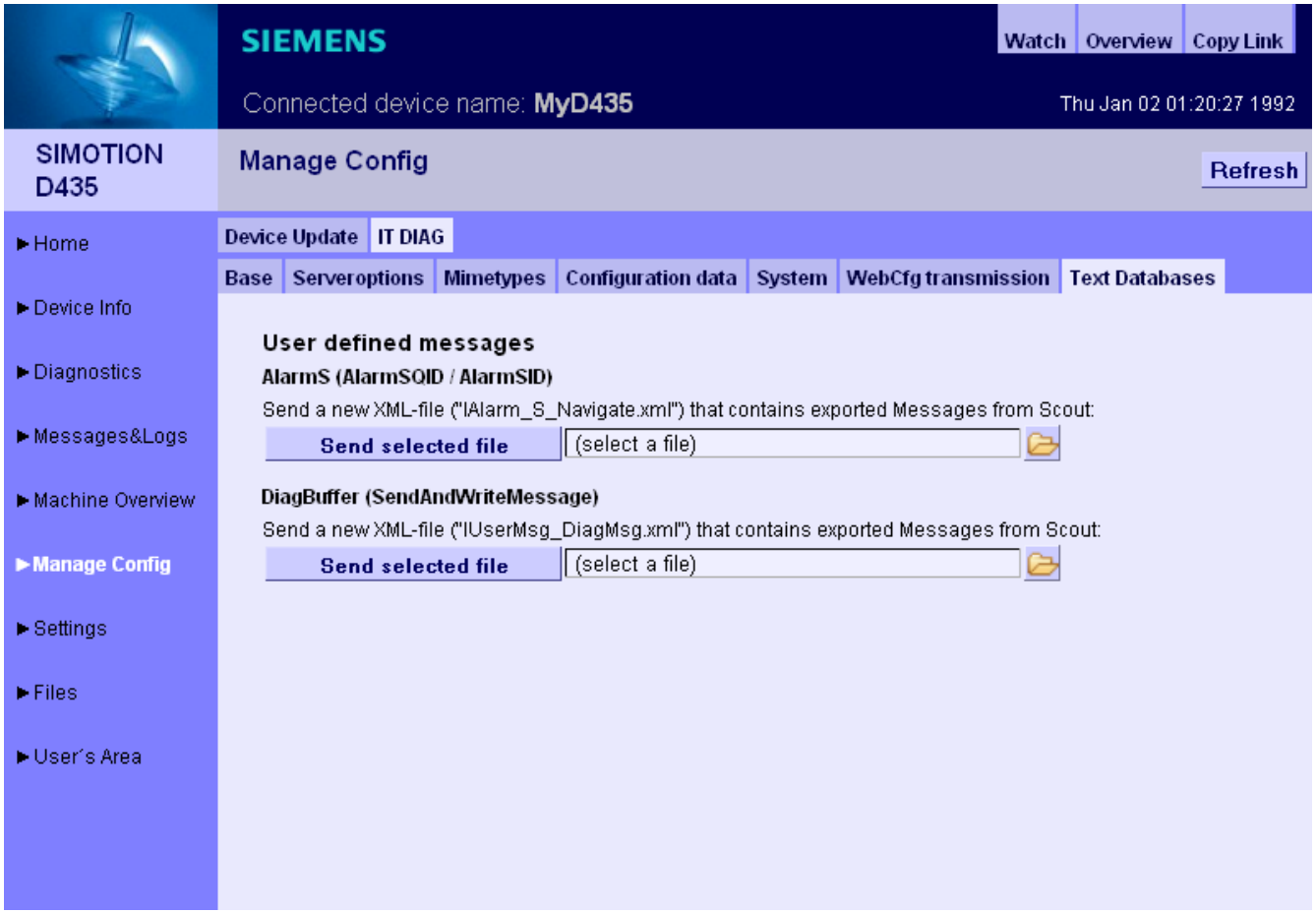


Bild 3-49 Text Databases

IT DIAG bietet auf dieser Seite die Möglichkeit benutzerdefinierte AlarmS- und DiagBuffer-Meldungen, die vorher im SIMOTION SCOUT exportiert wurden, auf das Gerät zu übertragen.

Wählen Sie für AlarmS die Datei IAlarm_S_Navigate.xml und für DiagBuffer die Datei IUserMsg_Navigate.xml eines SIMOTION SCOUT Sprachexports aus. Es besteht die Möglichkeit unterschiedliche Sprachen für AlarmS- und DiagBuffer-Meldungen auszuwählen.

Nach der Übertragung der Dateien auf das Gerät befinden sich die Meldungen in zwei Dateien

- dgusralarm.edb
- dgusrtxt.edb

im Verzeichnis /user/simotion/HMICFG. Diese Dateien können auf andere Steuerungen übertragen werden.

Sprachexport aus dem SIMOTION SCOUT

Im SIMOTION SCOUT ermöglicht der Menüpunkt **Projekt Meldungen** den Export benutzerdefinierter Meldungen.

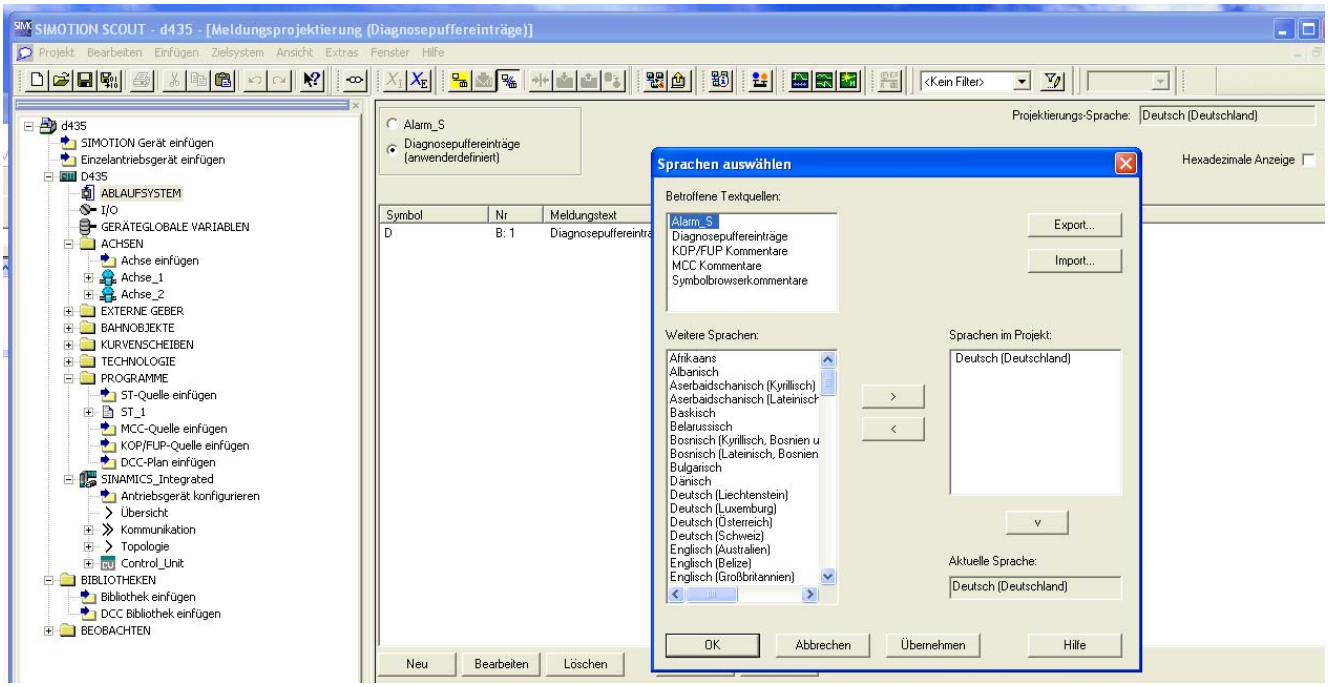


Bild 3-50 SIMOTION SCOUT Sprachenexport Sprachauswahl

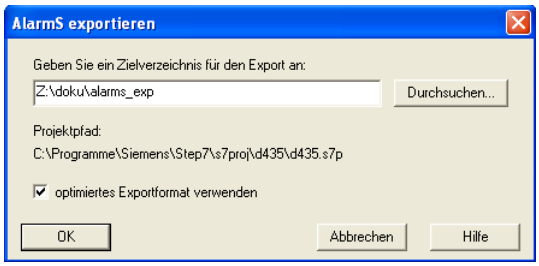


Bild 3-51 SIMOTION SCOUT Sprachenexport Angabe des Zielverzeichnisses

Beim Export werden alle benutzerdefinierten Texte in allen vorhandenen Sprachen in XML-Dateien exportiert. Beim Upload zum Gerät wird nur die im SIMOTION SCOUT voreingestellte Sprache gespeichert.

Jede Änderung im SIMOTION SCOUT bedingt den erneuten Export und Upload der Texte.

3.2.7 Settings

Diese Seite ermöglicht es, verschiedene Einstellungen zu ändern.

In den Bereichen **Operation state** und **Time Settings** können Einstellungen des SIMOTION Geräts geändert werden.

Im Bereich **User Pages** kann die Anzeige benutzerdefinierter Seiten und des **IT DIAG** Menüeditors verändert werden.

Hinweis

Die Seite **Settings** ist passwortgeschützt.

Zum Auslieferungszeitpunkt sind der User = simotion und das Passwort = simotion voreingestellt.

Sie sollten dieses Passwort unbedingt ändern, um Sicherheitsprobleme zu vermeiden.

Siehe Loginverwaltung (Seite 18)

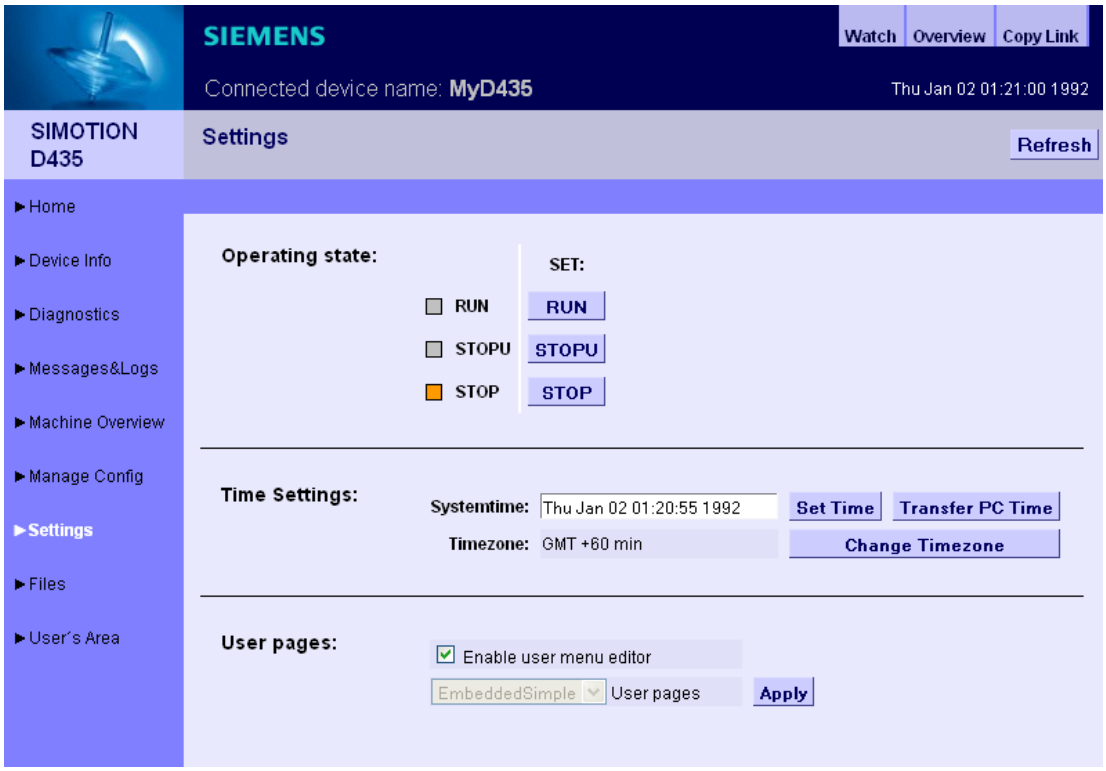


Bild 3-52 Settings

Zustand des SIMOTION Geräts ändern

Operation state

In dem Feld für den Betriebszustand des SIMOTION Geräts wird mit dem Drücken der jeweiligen Schaltfläche RUN, STOPU oder STOP die Anforderung zum Wechsel des Betriebszustandes ausgelöst.

Der Schalter auf der Steuerung hat gegenüber dieser Eingabe höhere Priorität, d. h., wenn dieser Schalter auf STOP steht, ist kein RUN möglich.

Hinweis: Der aktuelle Betriebszustand muss für eine Übertragung des Projekts oder der Firmware auf STOP gesetzt werden.

⚠ GEFAHR
Gefahren für Mensch und Maschine können durch nicht kontrollierten Wechsel des Betriebszustandes entstehen. Beachten Sie die Sicherheitsvorschriften, bevor Sie den Betriebszustand eines SIMOTION Geräts über diese Funktionen ändern.

Time Settings

In dem Feld für die Zeiteinstellungen wird für das SIMOTION Gerät die Systemzeit und die Zeitzone in Minuten inklusive Vorzeichen eingestellt.

Systemtime	Uhrzeit des SIMOTION Geräts am Standort
Timezone	Differenz zwischen der Systemtime am Standort und GMT

Die Systemzeit und Zeitzone sind für den OPC XML-DA Zugriff relevant.

Der OPC XML-DA Client erwartet alle vom SIMOTION Gerät gesendeten Zeitangaben als GMT. Ein SIMOTION Gerät ist aber auf Ortszeit (GMT + X) eingestellt, aus diesem Grund muss eine Zeitzone für das SIMOTION Gerät eingestellt werden.

Der Button **Change Timezone** öffnet eine Zeitzonenliste, aus der eine Zeitzone ausgewählt werden kann.

Mit Browsern, die die Listendarstellung nicht unterstützen, ist die Differenz in Minuten mit Vorzeichen im Bereich von -720 bis +780 einzugeben.

Die Zeitzone kann auch in der **Hardwarekonfiguration > Objekteigenschaften der CPU > Reiter "Ethernet erweitert" > OPC XML/Diagnoseseiten** eingestellt und über einen Download gesetzt werden.

User Pages

Die Checkbox **Enable user menu editor** ermöglicht die Anschaltung des Menüeditor-Links auf den benutzerdefinierten Seiten. Diese Option ist nur wirksam, wenn die Auswahl **Embedded** in der Auswahlbox **User Pages** gewählt wurde.

Die Auswahlbox **User Pages** beeinflusst die Art der Anzeige benutzerdefinierter Seiten. Siehe Eingebettete anwenderdefinierte Seiten (Seite 126)

3.2.8 Files

3.2.8.1 Files

Über die Seite **Files** können Sie auf der Speicherkarte im SIMOTION Gerät Unterverzeichnisse erstellen, anwählen und löschen. Weiterhin können Sie Dateien speichern, anzeigen und löschen.

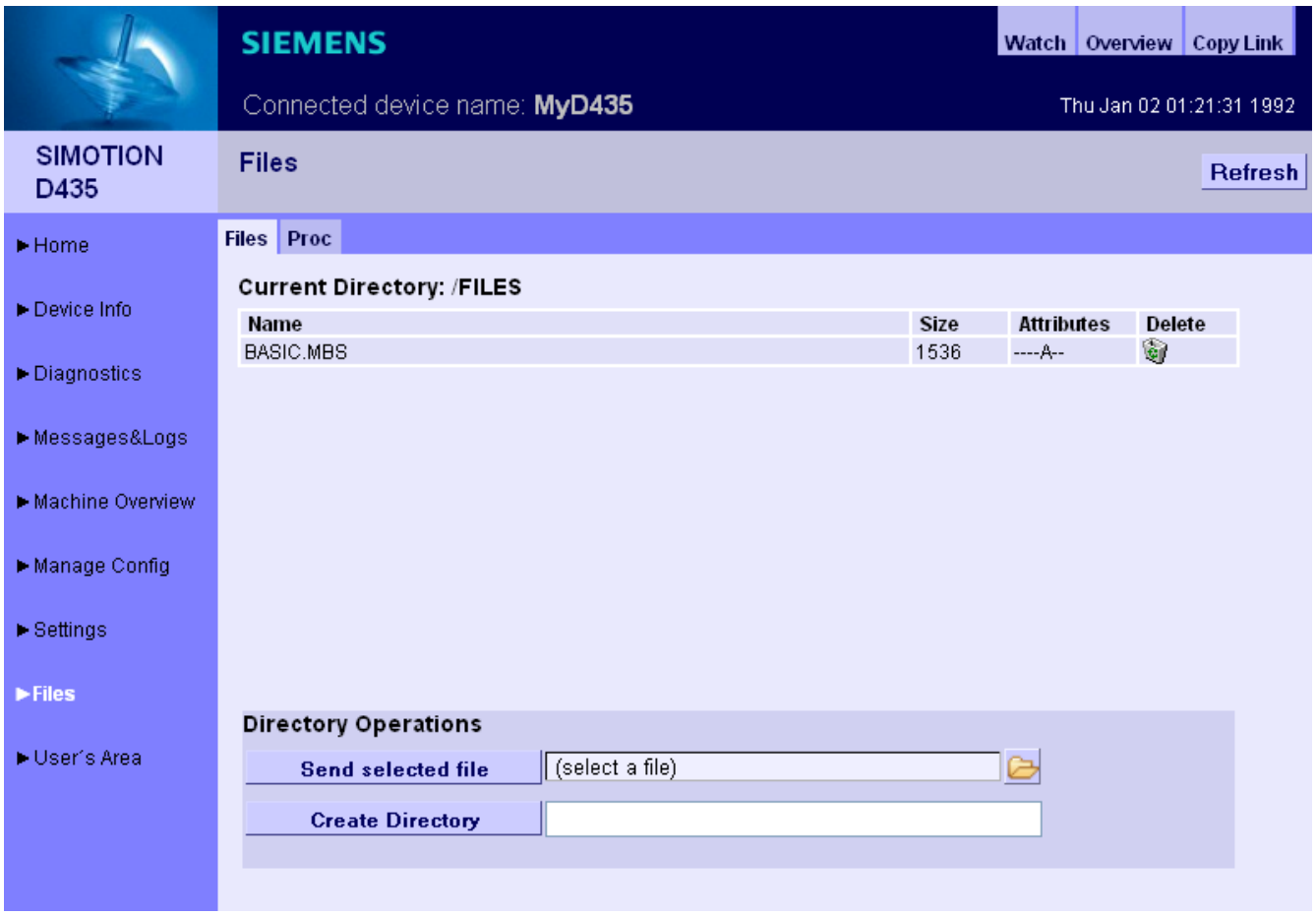


Bild 3-53 Files

Dateien und Verzeichnisse verwalten

Die Ablage der anwenderspezifischen Verzeichnisse und Dateien erfolgt in einem speziellen Verzeichnis. Bei einer Default-Installation sind dies:

SIMOTION Gerät	Pfad
C, D	\USER\SIMOTION\HMI\FILES
P350	F:\SIMOTION\USER\CARD\USER\SIMOTION\HMI\FILES
P320	D:\Card\USER\SIMOTION\HMI\FILES

Zur Erstellung von Unterverzeichnissen geben Sie im Eingabefeld den gewünschten Namen ein und bestätigen Sie anschließend durch Anklicken des Buttons **Create Directory**.

Über das Symbol Papierkorb können Sie Dateien und Verzeichnisse löschen. Beim Löschen von Verzeichnissen müssen Sie sicherstellen, dass sich keine Dateien in diesem Verzeichnis befinden. Ggf. müssen Sie die einzelnen Dateien zuvor löschen.

Hinweis

Den auf der Karte verfügbaren Speicherplatz können Sie auf der Diagnostics-Seite in der Zeile "Memory Card" erfahren (Diagnostics (Seite 29)).

Dateien auf die SIMOTION Steuerung kopieren

Der Button **Send selected file** ermöglicht den Transfer einer Datei aus dem lokalen Dateisystem in die SIMOTION Steuerung. Über den Button mit dem Ordnersymbol können Sie aus Ihrem lokalen Dateisystem eine Datei auswählen und mit dem **Send selected file**-Button in die SIMOTION Steuerung übertragen.

Hinweis

Beim Upload einer Datei, die mit selbem Namen bereits in der SIMOTION Steuerung abgespeichert ist, wird die vorhandene Datei überschrieben.

3.2.8.2 Proc

Zugriff auf die Gerätevariablen mit dem Proc-Dateisystem

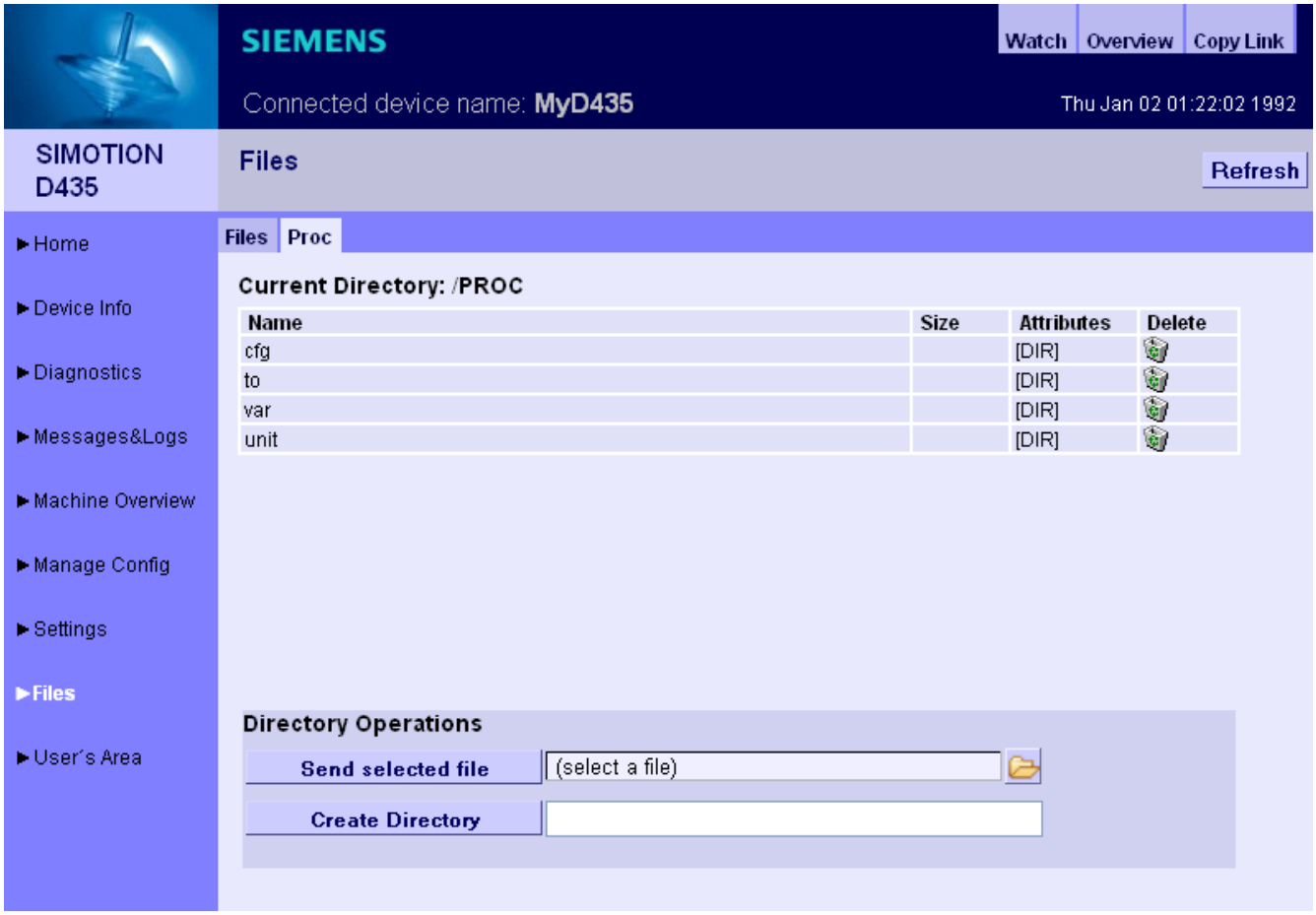


Bild 3-54 Proc-Dateisystem

Das Proc-Dateisystem bildet die Gerätevariablen als Laufwerk im Browser ab. Das ermöglicht z. B. das Auslesen von Gerätevariablen per FTP.

Der Variablenzugriff erfolgt über eine Pfadangabe und das Anhängen der Endung "bin" an den Variablennamen.

Variablen	Pfad
TO-Konfigurationsdaten	`/cfg/<toname>/<varname>.bin`
Systemvariablen von TOs	`/to/<toname>/<varname>.bin`
Gerätesystemvariablen	`/var/<varname>.bin`
Programmvariablen	`/unit/<unitname>/<varname>.bin`

Auch der Zugriff auf Arrays wird über einen Pfad realisiert.

- Variable: `unit/UnitName.StructName.StructCompSimple`
- Pfad：`/unit/UnitName/StructName/StructCompSimple.bin`

Zugriff auf Arrays und Strukturen

- Variable: `unit/UnitName.Array[5].StructName.StructCompSimple`
- Pfad: `/unit/UnitName/Array/5/StructName/StructCompSimple.bin`

Die Dateien des ProcFS enthalten den Variableninhalt in binärer Form in der Darstellung (Endianess) der verwendeten Steuerung.

3.3 Vereinfachte Standardseiten

3.3.1 BASIC Seiten

Darstellung von IT DIAG Seiten auf Geräten mit kleinem Display

Für die optimierte Darstellung von IT DIAG Seiten auf Geräten, wie Handy oder PDA, werden ab Version 4.1.3 spezielle Seiten bereitgestellt.

Für die Darstellung, der Basic Seiten von IT DIAG wird, folgende Mindestkonfiguration empfohlen:

- Mobiles Betriebssystem mit installiertem Web-Browser, der den HTML4-Standard unterstützt
- Bildschirmauflösung von mindestens 320x240 Pixeln und Farbdarstellung
- Touchscreen bzw. Zeigereingabegerät
- Zur Nutzung des vollen Funktionsumfangs wird JavaScript (ECMA-262) benötigt.

Einstiegspunkt für diese Seiten ist die Adresse http://<IPAddr>/BASIC

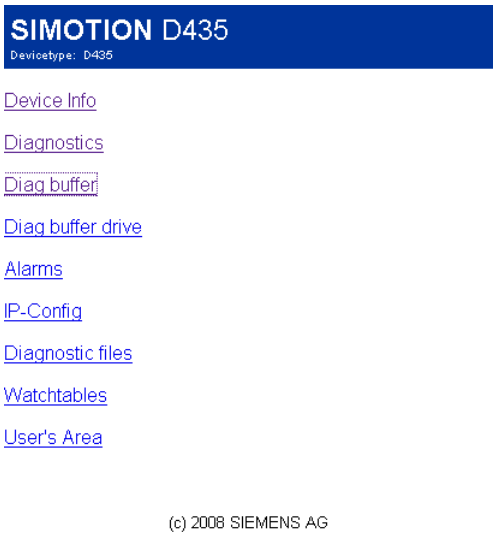


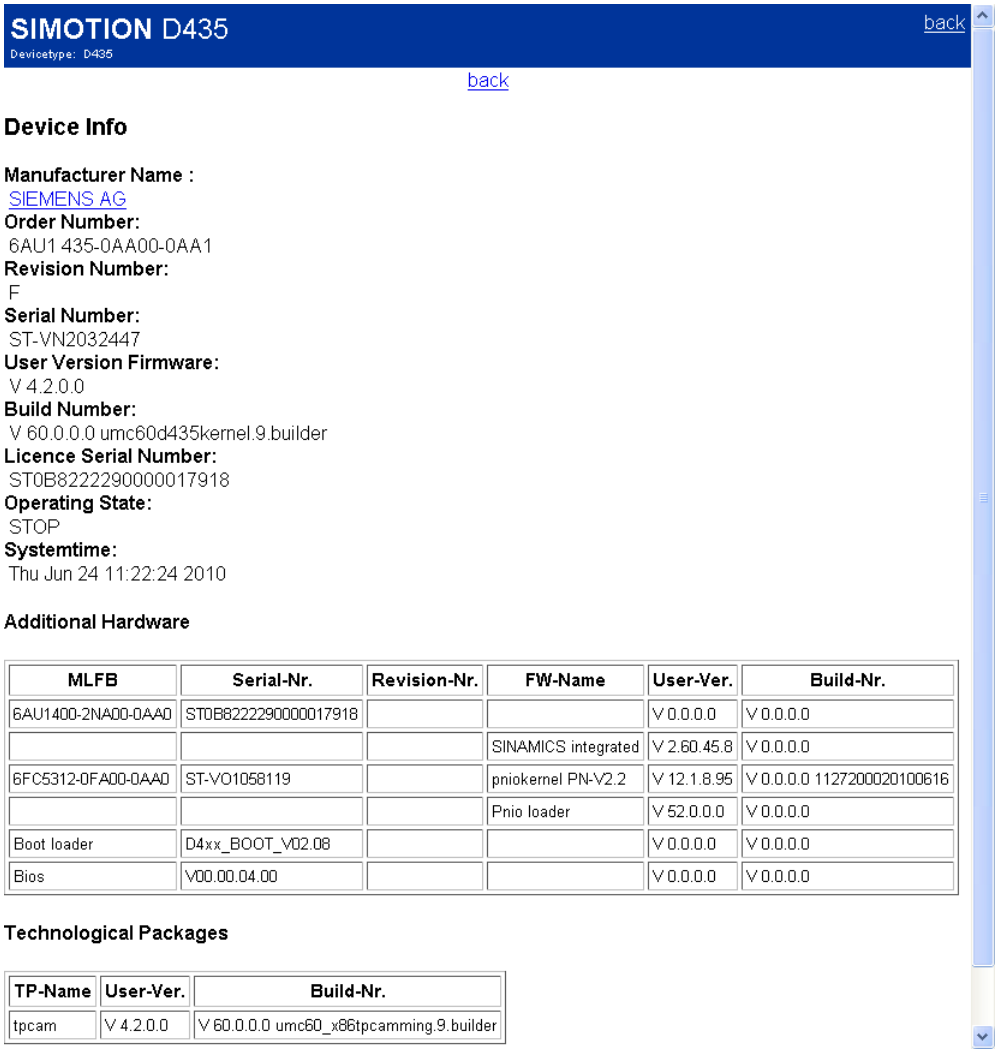
Bild 3-55 Startbildschirm der vereinfachten HTML-Seiten

3.3.2 Device Info

Hard- und Firmware-Informationen

Auf der Seite **Device Info** werden folgende aktuelle Hard- und Firmware-Informationen des SIMOTION Geräts angezeigt:

Manufacturer Name	Siemens AG
Order Number	Bestell-Nummer (MLFB) des Geräts
Revision Number	Hardwareversion
Serial Number	Seriennummer des SIMOTION Geräts
User Version Firmware	SIMOTION Kernel Anwender Version
Build Number	interne Versionsnummer
Additional Hardware	Gesteckte Komponenten des SIMOTION Geräts mit:
	MLFB, Seriennummer, Revisionsnummer Firmwarenamen, Anwender Versionsnummer, interne Versionsnummer
Technological Packages	Geladene Technologiepakete mit:
	Paketname, Anwender Versionsnummer, interne Versionsnummer



SIMOTION D435
Devicetype: D435

back

back

Device Info

Manufacturer Name :
SIEMENS AG
Order Number:
6AU1 435-0AA00-0AA1
Revision Number:
F
Serial Number:
ST-VN2032447
User Version Firmware:
V 4.2.0.0
Build Number:
V 60.0.0.0 umc60d435kernel.9.builder
Licence Serial Number:
ST0B8222290000017918
Operating State:
STOP
Systemtime:
Thu Jun 24 11:22:24 2010

Additional Hardware

MLFB	Serial-Nr.	Revision-Nr.	FW-Name	User-Ver.	Build-Nr.
6AU1400-2NA00-0AA0	ST0B8222290000017918			V 0.0.0.0	V 0.0.0.0
			SINAMICS integrated	V 2.60.45.8	V 0.0.0.0
6FC5312-0FA00-0AA0	ST-VO1058119		pniokernel PN-V2.2	V 12.1.8.95	V 0.0.0.0 1127200020100616
			Pnio loader	V 52.0.0.0	V 0.0.0.0
Boot loader	D4xx_BOOT_V02.08			V 0.0.0.0	V 0.0.0.0
Bios	V00.00.04.00			V 0.0.0.0	V 0.0.0.0

Technological Packages

TP-Name	User-Ver.	Build-Nr.
tpcam	V 4.2.0.0	V 60.0.0.0 umc60_x86tpcamming.9.builder

Bild 3-56 Device Info auf vereinfachten HTML-Seiten

3.3.3 Diagnostics

Übersicht über den allgemeinen Zustand der SIMOTION Steuerung

Die **Diagnostics**-Seite zeigt die folgenden Zustände der SIMOTION Steuerung an:

Systemtime	Aktuelle Uhrzeit der SIMOTION Steuerung
Timezone	Aktuelle Differenz zwischen der Systemtime und GMT in Minuten
CPU Load by cyclic Tasks	Prozentualer Rechenzeitanteil der Servo und IPO Ebenen an der Gesamtrechenzeit
Memory Load	Größe und Belegung des Speichers, der RAM-Disk, der Speicherkarte und des netzausfesten Speichers in Bytes
State	Aktueller Betriebszustand der SIMOTION Steuerung

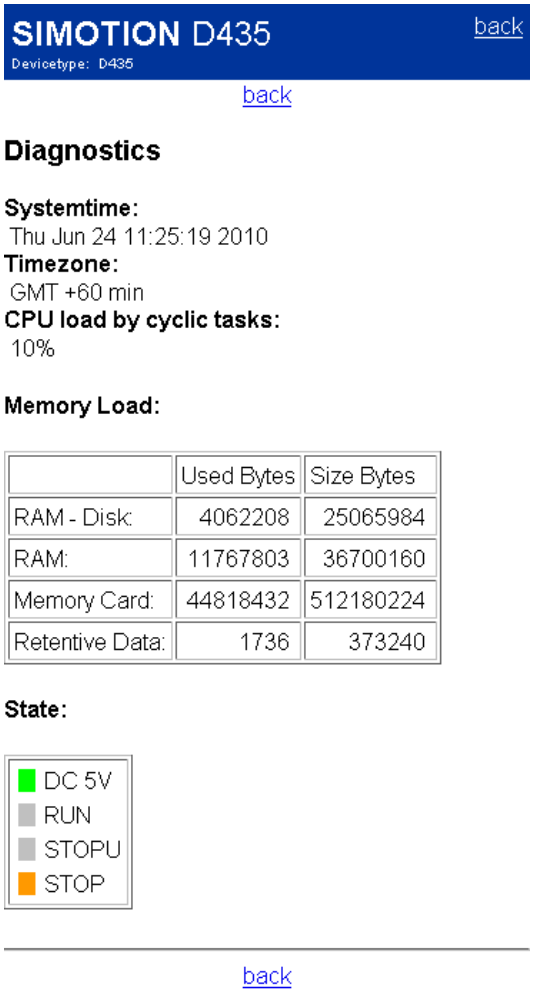


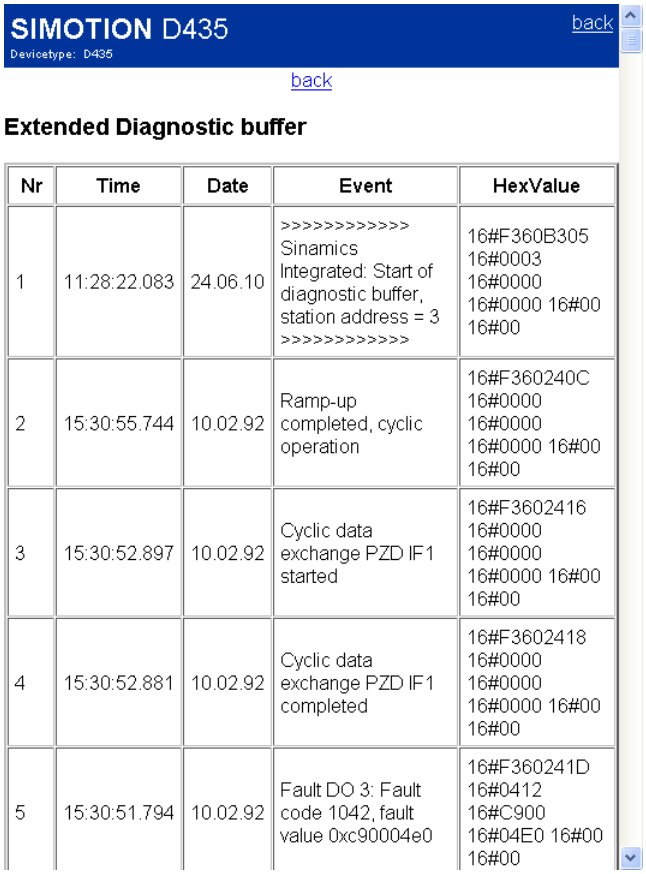
Bild 3-57 Diagnostics auf vereinfachten HTML-Seiten

3.3.4 Diag buffer

Diag buffer Informationen

Die **Diag buffer**-Seite stellt die Ereignisse des Diagnosepuffers dar.

Time	Zeitpunkt des Ereignisses
Date	Datum des Ereignisses
Event	Anzeige des Ereignisses als Text. Bei fehlender Sprachdatei DGBUFTXT.EDB erfolgt die Anzeige in Hexadezimaldarstellung
HexValue	Hex-Wert der Diagnosepuffermeldung



Nr	Time	Date	Event	HexValue
1	11:28:22.083	24.06.10	>>>>>>>>>>>> Sinamics Integrated: Start of diagnostic buffer, station address = 3 >>>>>>>>>>>>	16#F360B305 16#0003 16#0000 16#0000 16#00 16#00
2	15:30:55.744	10.02.92	Ramp-up completed, cyclic operation	16#F360240C 16#0000 16#0000 16#0000 16#00 16#00
3	15:30:52.897	10.02.92	Cyclic data exchange PZD IF1 started	16#F3602416 16#0000 16#0000 16#0000 16#00 16#00
4	15:30:52.881	10.02.92	Cyclic data exchange PZD IF1 completed	16#F3602418 16#0000 16#0000 16#0000 16#00 16#00
5	15:30:51.794	10.02.92	Fault DO 3: Fault code 1042, fault value 0xc90004e0	16#F360241D 16#0412 16#C900 16#04E0 16#00 16#00

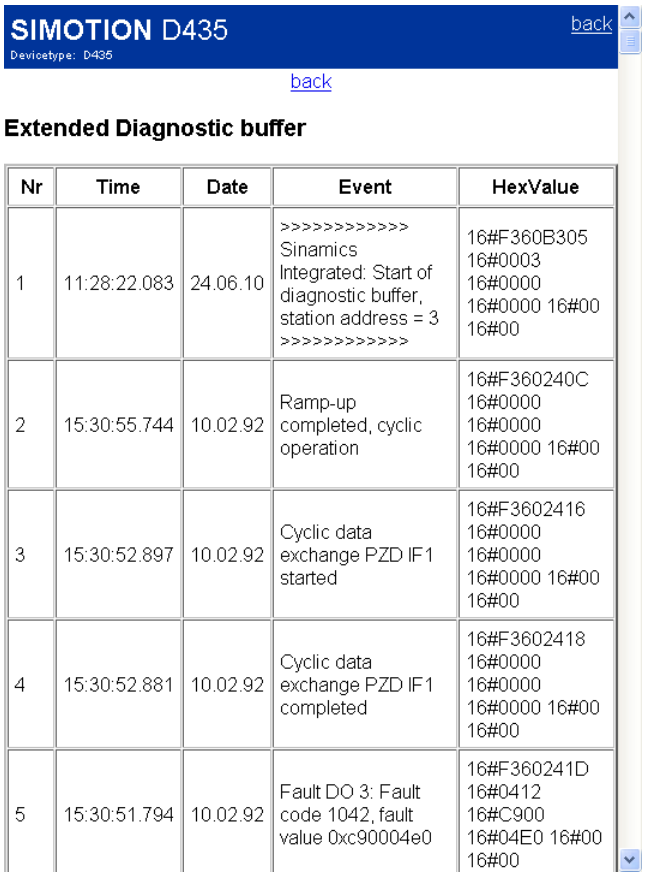
Bild 3-58 Diagnosepuffer in vereinfachter Darstellung

3.3.5 Diag buffer drive

Diag buffer drive Informationen

Die **Diag buffer drive**-Seite stellt die Ereignisse des Antriebsdiagnosepuffers für die integrierten Antriebe dar.

Time	Zeitpunkt des Ereignisses
Date	Datum des Ereignisses
Event	Anzeige des Ereignisses als Text. Bei fehlender Sprachdatei DGEXTXT.EDB erfolgt die Anzeige in Hexadezimaldarstellung
HexValue	Hex-Wert der Antriebsdiagnosepuffermeldung



Nr	Time	Date	Event	HexValue
1	11:28:22.083	24.06.10	>>>>>>>>>>>> Sinamics Integrated: Start of diagnostic buffer, station address = 3 >>>>>>>>>>>>	16#F360B305 16#0003 16#0000 16#0000 16#00 16#00
2	15:30:55.744	10.02.92	Ramp-up completed, cyclic operation	16#F360240C 16#0000 16#0000 16#0000 16#00 16#00
3	15:30:52.897	10.02.92	Cyclic data exchange PZD IF1 started	16#F3602416 16#0000 16#0000 16#0000 16#00 16#00
4	15:30:52.881	10.02.92	Cyclic data exchange PZD IF1 completed	16#F3602418 16#0000 16#0000 16#0000 16#00 16#00
5	15:30:51.794	10.02.92	Fault DO 3: Fault code 1042, fault value 0xc90004e0	16#F360241D 16#0412 16#C900 16#04E0 16#00 16#00

Bild 3-59 Diag Buffer Drive

3.3.6 Alarms

Informationen zu Alarmen

Level	Kategorie des Alarms
Time	Zeitpunkt des Alarms
TO	Technologieobjekt, das den Alarm ausgelöst hat
Nr	Alarmnummer
Text	Anzeige der Alarmmeldung als Text

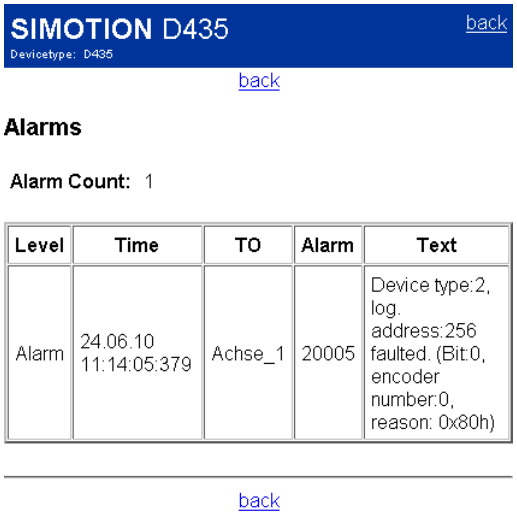


Bild 3-60 Alarms in vereinfachter Darstellung

3.3.7 IP-Config

Daten der Ethernet-Schnittstelle der SIMOTION Steuerung

IP Address	Adresse der TCP/IP Schnittstelle
Subnet Mask	Subnet Mask der Schnittstelle
MAC Address	Adresse der Netzwerkkarte
Gateway	Defaultgateway der Schnittstelle
	Die Angabe steht immer in der ersten Spalte. Sie steht nicht unbedingt im Zusammenhang mit der IP-Adresse der Spalte, sondern kann auch bei den weiteren Schnittstellen projektiert worden sein.

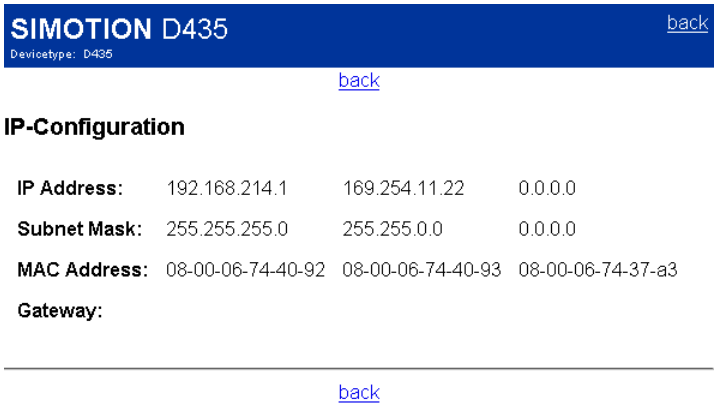


Bild 3-61 IP-Config

3.3.8 Diagnostic Files

Diagnoseseiten des Webservers sichern

Die allgemeinen Diagnosedaten und einzelne HTML-Seiten von IT DIAG können über diese Seite gesichert werden.

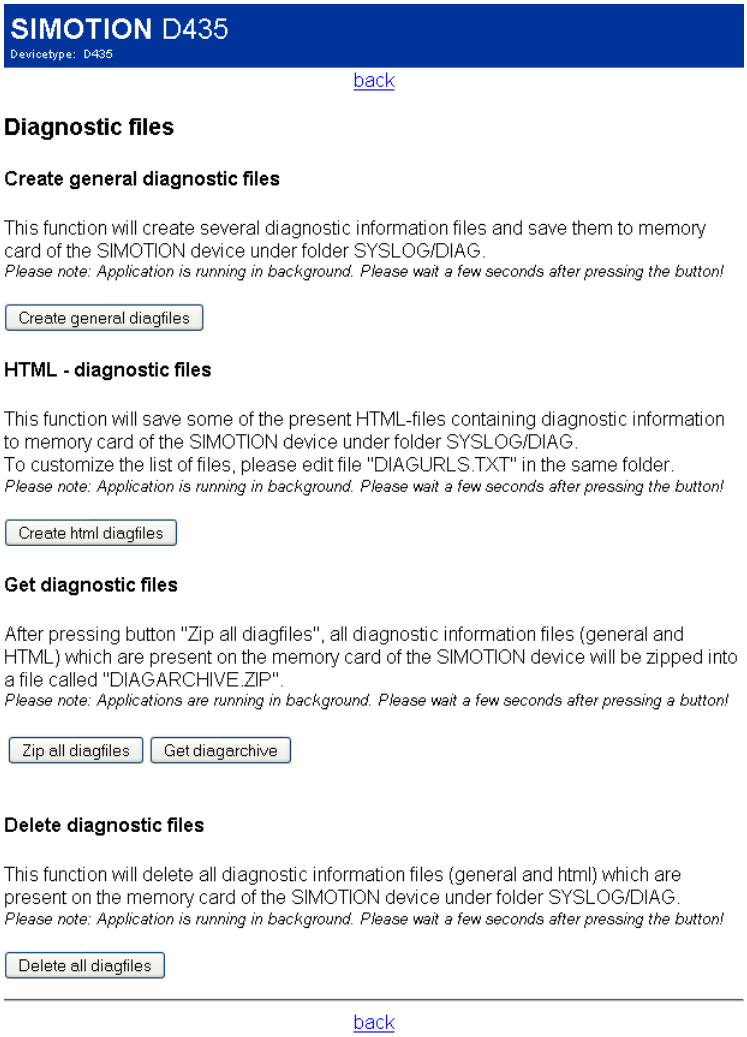


Bild 3-62 Diagnostic Files

Siehe auch

Diagnostic files (Seite 51)

3.3.9 Watchtables

Watchtables

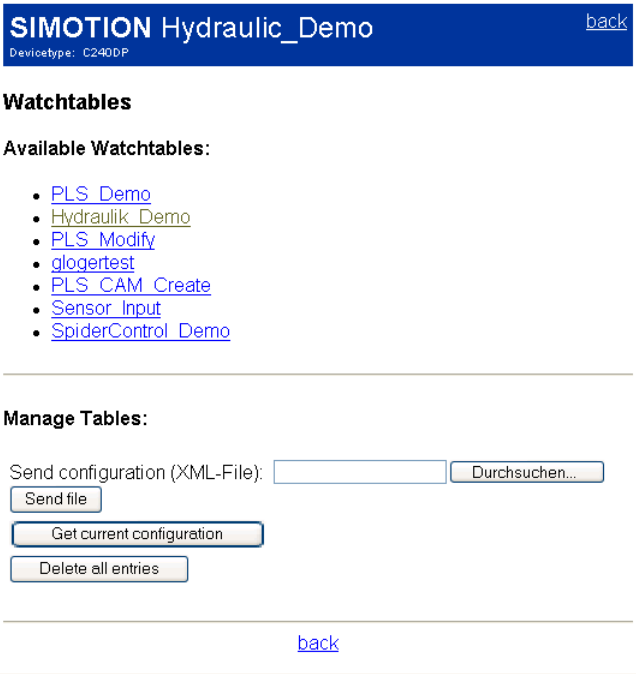


Bild 3-63 Watchtables

Diese Seite zeigt alle eingerichteten Watchtables. Diese Watchtables sind dieselben, wie auf der Standard IT DIAG Seite. Sie können gespeichert, gelöscht und hochgeladen werden. Eine Editierung ist an dieser Stelle nicht möglich.

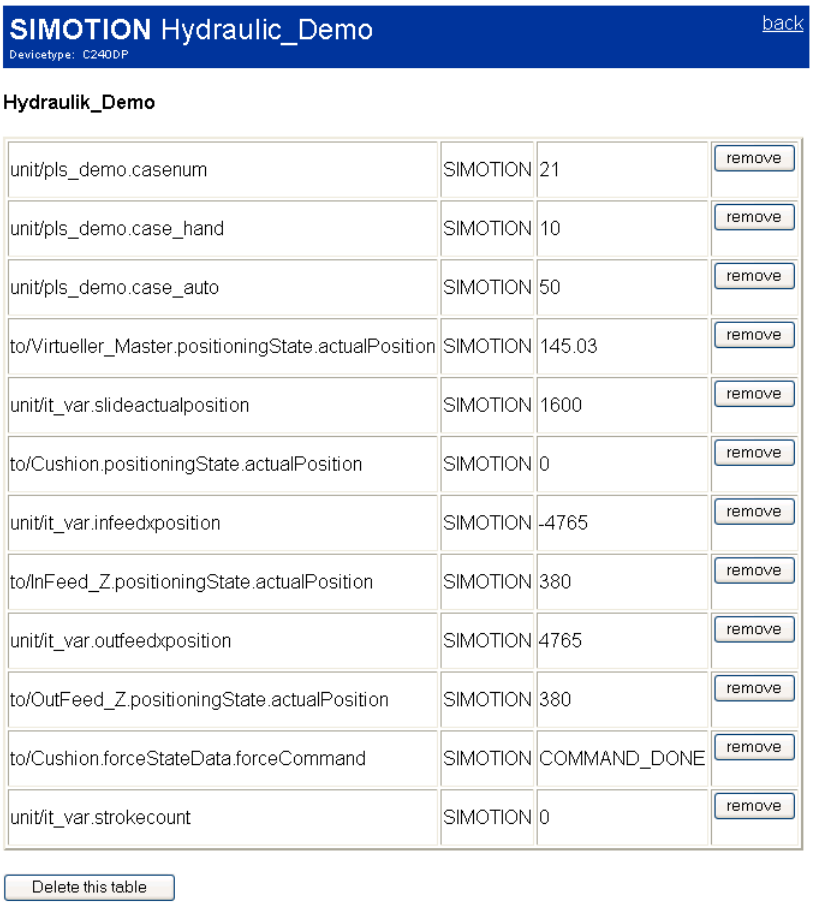


Bild 3-64 Anzeige einer Watchtable

Siehe auch

Watch (Seite 35)

3.3.10 User's Area

User's Area

Bild 3-65 User's Area

In der User's Area werden benutzerdefinierte Seiten dargestellt.

3.4 IT DIAG Konfiguration

3.4.1 Einleitung

Es gibt zwei Dateien, mit denen IT DIAG konfiguriert werden kann:

- WebCfg.xml
- WebCfgFrame.xml

Über die Konfigurationsdatei WebCfg.xml werden anwenderrelevante Einstellungen im Webserver konfiguriert. Die WebCfgFrame.xml enthält die IT DIAG Einstellungen des Herstellers.

WebCfg.xml

Die Datei ist in einzelne Sektionen gegliedert, z. B. Serveroptionen und Benutzerdatenbank. Die WebCfg.xml kann während der Laufzeit neu geladen werden. Dies führt zu einem Neustart des Webservers. Nach dem Neustart stehen Ihnen die geänderten Einstellungen zur Verfügung.

Über die Standardseiten **Manage Config > IT DIAG** können Einträge in der WebCfg.xml auf sichere Art und Weise geändert werden. IT DIAG Base (Seite 76)

Die Konfigurationsdatei gliedert sich in verschiedene Bereiche:

- Virtuelles Datei System: Abbildung eines hierarchischen Dateisystems über die Konfiguration.
- Server Optionen: Austauschen der Startseite der Diagnose-Standardseiten durch eine eigene Startseite (siehe Anwenderdefinierte Startseite (Seite 125)), Porteinstellungen.
- Konfigurationsbereich: modulspezifische Konfigurationsdaten
- Benutzerdatenbank: Steuerung der Zugriffe auf die Diagnoseseiten (siehe Kapitel Loginverwaltung (Seite 18)).
- Dateitypen: Festlegung des Mime-Type (Seite 278) im HTTP-Header.

Die Datei WebCfg.xml finden Sie entweder auf der Liefer-DVD im Verzeichnis 3_Configuration (im Default Zustand) oder auf der Speicherkarte der SIMOTION Steuerung im Verzeichnis USER\SIMOTION\HMICFG\.

WebCfgFrame.xml

Die WebCfgFrame.xml enthält die IT DIAG Grundeinstellungen des Geräts.

ACHTUNG
Nehmen Sie nur die in diesem Dokument beschriebenen Einstellungen vor. Änderungen von in diesem Handbuch nicht beschriebenen Einstellungen in der WebCfg.xml und WebCfgFrame.xml können Fehler verursachen.

3.4.2 Überblick

An dieser Stelle wird ein Überblick über die zur Verfügung stehenden Konfigurationsmöglichkeiten gegeben. Im Anhang befinden sich weitere Erläuterungen und Beispiele zu den einzelnen Optionen.

Das virtuelle Dateisystem und das <DEFAPP>-Tag werden aufgrund ihrer Komplexität auf den nächsten Seiten ausführlicher behandelt.

WebCfg

Tabelle 3- 3 Übersicht WebCfg.xml

Tag	Wert/Typ	Bemerkung
<ALTERNATE_PORTNUMBER> (Seite 273)	[Portnummer]: Ganzzahl	Alternative Portnummer
<ALTERNATE_SSL_PORTNUMBER > (Seite 274)	[Portnummer]: Ganzzahl	Alternative SSL Portnummer
<BASE> (Seite 274)		Linklisten aller HTML-Seiten
<CONFIGURATION_DATA> (Seite 276)		Dieses Tag umschließt die anderen Konfigurationsdaten
<MIME_TYPES> (Seite 278)	[<FILE EXTENSION> - Tags]: Text	Zuweisung von Dateitypen zu Dateiendungen
<SERVEROPTIONS> (Seite 279)		Beinhaltet die Serveroptionen
<TIMEZONE> (Seite 280)	[-+Minuten]: vorzeichenbehaftete Ganzzahl	Zeitzonensynchronisation
<USERDATABASE> (Seite 281)		Verschiedenste Bereiche des Webservers, angefangen von HTML-Seiten, Verzeichnissen etc. bis hin von einzelnen Aktionen von Applikationen können mit einem Zugriffsschutz geschützt werden.

Tabelle 3- 4 <SERVEROPTIONS>

Tag	Bemerkung
<BROWSEABLE> (Seite 275)	Browsen in einem Verzeichnis oder global erlauben oder verbieten
<DEFAULTDOCUMENT> (Seite 277)	Dokument, dass ausgegeben wird, wenn in der URL kein Dokument enthalten ist.
<PORTNUMBER> (Seite 279)	Portnummer des TCP/IP Server. Default : 5001
<SSLPORTNUMBER> (Seite 280)	Portnummer des SSL Port. Default : 5443

WebCfgFrame

Tabelle 3- 5 Übersicht WebCfg.xml

Tag	Bemerkung
<BASE> (Seite 287)	Linklisten der Standardseiten (HTML bzw. XML)
<CONVERSION> (Seite 288)	1. Sendet Daten 2. Empfängt die WebCfg.xml 3. Generiert den Verzeichnis-Browser
<DEFAPP> (Seite 289)	Default Applikation
<HTTP_RESULT_CODES> (Seite 291)	Anpassung der HTTP-Fehlermeldungen
<SSL> (Seite 293)	Enthält einen Eintrag für die Verschlüsselung der SSL-Verbindung

3.4.3 Konfiguration des Dateisystems

3.4.3.1 Virtuelles Dateisystem

Der Webserver ist in erster Linie für embedded Geräte entwickelt worden. Hierbei kann man nicht zwingend davon ausgehen, dass diese Zielsysteme über Dateisysteme, wie Speicherkarten oder Festplatten verfügen.

Das Adressierungsmodell, das im Internet verwendet wird, (URLs) setzt aber ein hierarchisches Dateisystem auf dem Server voraus.

Da XML-Dateien ebenfalls eine hierarchische Struktur aufweisen, lag es Nahe, eine XML-Datei zu verwenden, um die geforderte Datei-Hierarchie in einer Datei (bzw. Speicherblock) abzubilden.

Das war die grundlegende Idee, warum der Webserver über die XML-Konfigurationsdatei WebCfg.xml verfügt. Des Weiteren hat es sich angeboten, auch andere Konfigurationsparameter mit in dieser Datei abzulegen, sodass es jetzt eine zentrale Stelle gibt, mit der der Webserver nebst seinen Modulen konfiguriert wird. Diese Datei kann auch automatisiert erstellt und geladen werden, z. B. wenn ein CS-System (Wie SIMOTION SCOUT, STEP7, etc.) den Webserver in einem Zielsystem zusammen mit anderen Konfigurationsparametern des Zielsystems konfigurieren soll.

Bei jeder Anforderung eines Clients wird als Erstes im "XML–Dateisystem" nach einem passenden Eintrag (Tag) oder Link gesucht. Alle Sicherheitseinstellungen etc. werden nur im "XML–Dateisystem" festgelegt.

Aufbau des XML Dateisystems innerhalb der XML-Konfigurationsdatei:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE LOCALLINK="link_zu_externen_filesystem">
      <www LOCALLINK="/" BROWSEABLE="TRUE"
           REALM="FileAdministrator"/>
      <Test.mcs LINK="/Tests/Test.mcs/"/>
      <Default.mcs>
         <![CDATA[
         <HTML>
            <HEAD>
            [...]
         ]]>
       </Default.mcs>
   </BASE>
   [...]
</SERVERPAGES>
```

3.4.3.2 Virtuelles vs. physikalisches Dateisystem

Wie im Abschnitt 'Virtuelles Dateisystem' bereits erläutert wurde, wird jeder Zugriff auf eine Datei erst im XML-Dateisystem gesucht.

Folgende Fälle können hierbei auftreten:

1. Es wird ein Link in das physikalische Dateisystem gefunden. Die Datei wird vom physikalischen Dateisystem geladen und zum Client zurückgesandt.
2. Die gesuchte Datei ist im XML-Dateisystem vorhanden, dann wird sie an den anfordernden Client zurückgesandt.

Die beiden Fälle des Dateizugriffs in grafischer Darstellung:

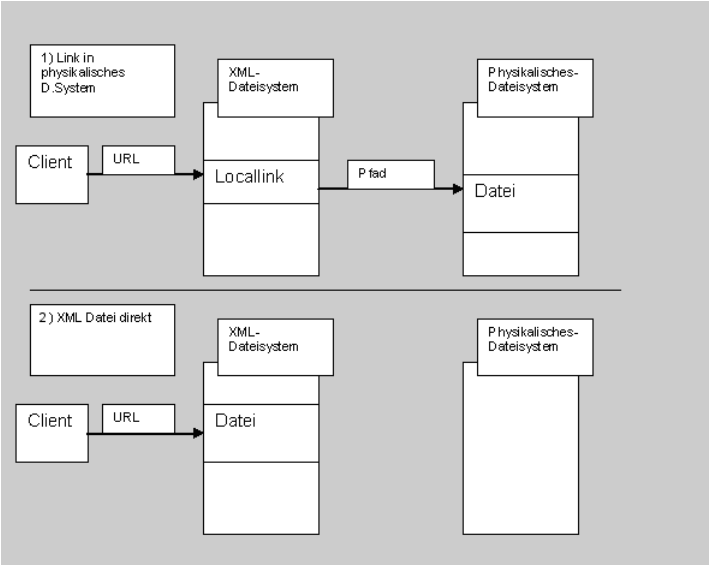


Bild 3-66 Mögliche Zugriffe über XML-Dateisystem

Hinweis

Ein Link kann auch ein Link auf ein Verzeichnis sein, nicht nur auf eine Datei. Wenn während des Auflösens der URL ein Link gefunden wird, wird dieser durch sein Ziel ersetzt und dann weiter gesucht.
Für die Sicherheitseinstellungen des externen Dateisystems gelten die des letzten gefundenen XML-Knotens, also des Knotens, dessen LOCALLINK Attribut die Suche im externen Dateisystem ausgelöst hat.

3.4.3.3 Externes Dateisystem vorrangig setzten <PREFER_EXTERNAL>

Das Attribut PREFER_EXTERNAL kann nur in Verbindung mit LOCALLINK gesetzt werden.

Ist PREFER_EXTERNAL gesetzt und existiert eine Datei im XML und im externen Dateisystem, wird die Datei aus dem externen Dateisystem bevorzugt.

Wenn dieses Attribut nicht gesetzt ist, dann wird die Datei aus dem XML-System bevorzugt.

Gleiches gilt für Verzeichnisse (im Schreibfall).

3.4.3.4 Links in das physikalische Dateisystem <LOCALLINK>

Lokale Links sind die einzige Möglichkeit, um auf das physikalische Datei System zuzugreifen.

Jeder Datenknoten des XML-Dateisystems kann ein LOCALLINK Attribut besitzen, auch der <BASE> Knoten. Der <BASE> Knoten entspricht dem Root-Eintrag des Dateisystems. Wenn ein LOCALLINK Attribut gefunden wurde, so wird trotzdem weiterhin versucht, das XML-Dateisystem zu parsen. Das XML-Dateisystem hat Vorrang vor dem externen Dateisystem. Diese Regel gilt für lesende und schreibende Zugriffe.

Diese Priorität kann mit dem Attribut PREFER_EXTERNAL geändert werden, sodass das externe Dateisystem Vorrang vor dem XML-Dateisystem hat.

Hinweis

Unter Umständen können bereits im Dateisystem vorhandene Dateien durch diese Vorrangigkeit überdeckt werden.

Wird beim weiteren XML parsen ein weiteres LOCALLINK Attribut gefunden, so gilt immer das zuletzt gefundene.

Es gilt zu beachten, dass die Suche im physikalischen Dateisystem ab $WWWRoot erfolgt. $WWWROOT bezeichnet den Basispfad WWWRoot, zu dem relativ der Zugriff in das externe Dateisystem erfolgt.

Links auf Verzeichnisse müssen immer mit einem Hierarchietrennzeichen abgeschlossen werden ("/").

Beispiele für den Zugriff auf das Dateisystem:

URL	Ziel im physikalischen Dateisystem	Eintrag in WebCfg.xml	Bemerkung
/Default.mcs	$WWWROOT/Default.mbs	<Default.mbs LOCALLINK="$WWWROOT/Default.mbs" PREFER_EXTERNAL="TRUE"/>	Verdeckt eine Datei aus dem XML-Verzeichnis. Siehe oben Beispiel WebCfg.xml.
/NewFile.mcs	$WWWROOT/NewFile.mbs	<NewFile.mbs LOCALLINK="$WWWROOT/NewFile.mbs" PREFER_EXTERNAL="TRUE"/>	Macht eine Datei aus dem physikalischen Dateisystem zugänglich
/index.mcs	$WWWROOT/NewDir/index.mbs	<index.mbs LOCALLINK="$WWWROOT/NewDir/index.mbs" PREFER_EXTERNAL="TRUE"/>	Verdeckt eine Standardseite
/XMLDir/NewFile.mcs	Kindknoten von <XMLDir> im XML-Dateisystem		Fügt im XML - Dateisystem einen Knoten hinzu

Die Dateien im physikalischen Dateisystem müssen die Endung .mbs haben, ansonsten erhält man eine Fehlermeldung vom Webserver. Im Browser werden diese Seiten mit der Endung .mcs ausgegeben. Dies gilt auch für verlinkte Seiten.

3.4.3.5 Browsen von Verzeichnissen

Das Browsen (Durchsuchen) von Verzeichnissen kann aktiviert oder deaktiviert werden.

Dies wird über das Attribut `BROWSEABLE` geregelt. Ist das Attribut `TRUE`, so wird eine Verzeichnisansicht erlaubt.

Das Browsen von Verzeichnissen kann standardmäßig erlaubt werden, wenn der Wert von `BROWSEABLE` auf `TRUE` gesetzt ist.

Für das folgende Beispiel wird vorausgesetzt, dass WWWRoot auf ein Verzeichnis mit folgender Struktur zeigt:

```
/
 /Datei1

 /Directory1/
 /Directory1/Datei2.mcs
 /Directory1/Datei3.mcs
 /Directory1/Directory2

 /Datei4
```

WebCfg.xml:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE LOCALLINK="/">
      <www LOCALLINK="/" BROWSEABLE="TRUE" .../>
   </BASE>
   [...]
</SERVERPAGES>
```

Der Client fordert die URL `http://Servername/www/Directory1` an.

Der Parser sucht im XML-Dateisystem nach www im Root-Verzeichnis und findet `LOCALLINK="/"`.

Der Parser sucht im physikalischen Dateisystem nach `/Directory1`. Der Slash "/" in diesem Pfad bleibt erhalten, da im Tag `LOCALLINK="/"` vereinbart wurde. `Directory1` bezeichnet dann den Pfad.

Das Verzeichnis `Directory1` existiert im physikalischen Dateisystem. Da `Browseable = TRUE` ist und keine Default HTML-Seite angegeben wurde, wird die Browse-Ansicht des Verzeichnisses zurückgegeben.

Siehe auch

<BROWSEABLE> (Seite 275)

<DEFAULTDOCUMENT> (Seite 277)

3.4.3.6 Sicherheitskonzept

An jedem XML-Knoten des XML-Dateisystems können Rechteinformationen in Form von Attributen hinterlegt werden:

- `REALM` (Sicherheitsbereich)
- `READ` (Leserechte)
- `WRITE` (Schreibrechte)
- `MODIFY` (Modifikationsrechte)

`REALM` darf nur einen Gruppennamen enthalten, `READ`, `WRITE`, `MODIFY` eine Liste von Gruppennamen, die mit "," getrennt werden. Es dürfen keine Leerzeichen oder andere Whitespace Zeichen verwendet werden.

Jedem User ist eine Menge von Usergruppen zugeordnet.

Wird eine Datei von einem User angefordert, wird das XML-Dateisystem nach dieser Datei durchsucht. Hierbei wird entsprechend des Pfades der Datei der XML-Baum durchlaufen. Werden hierbei mehrere XML-Knoten durchlaufen, so muss der angemeldete User allen Rechten, aller "berührten" Knoten genügen.

Beispiel:

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE LOCALLINK="/">
      <MainDir REALM="USER" LOCALLINK="/Base/" >
         <www LOCALLINK="/WebSeiten/"
              BROWSEABLE="TRUE"
              READ="Administrator"
              WRITE="FileAdministrator" />
      </MainDir>
      <Test.mcs LOCALLINK="/Tests/Test.mcs/"/>
      <XMLDir>
      </XMLDir>
      <Default.mcs>
         <![CDATA[
         <HTML>
            <HEAD>
            [...]
         ]]>
      </Default.mcs>
   </BASE>
[...]
</SERVERPAGES>
```

Tabelle 3- 6 Arten von Dateirechten

URL	Zugriff	Gruppen	Bemerkung
/<Datei>.mcs	lesend	keine	
/<Datei>.mcs	schreibend	keine	Zugriff ist nicht erlaubt
/MainDir/<Datei>.mcs	lesend	USER	Login-Maske, wenn Gruppe USER nicht vorhanden ist

3.4.3.7 REALM

Sicherheitsbereich einrichten

Mit Realm wird im WWW-Umfeld ein Sicherheitsbereich bezeichnet. Soll ein Verzeichnis betreten werden, und der User ist nicht Mitglied des angegebenen Realms (oder der Benutzer ist noch nicht angemeldet), dann erscheint eine Aufforderung sich anzumelden (Authentication required).

Wird auf eine Datei zugegriffen, die über `REALM` geschützt ist, muss der Client sich authentifizieren. Web Browser blenden in der Regel eine Benutzer-Abfragebox auf, in die der Benutzer Username und Passwort angeben muss:

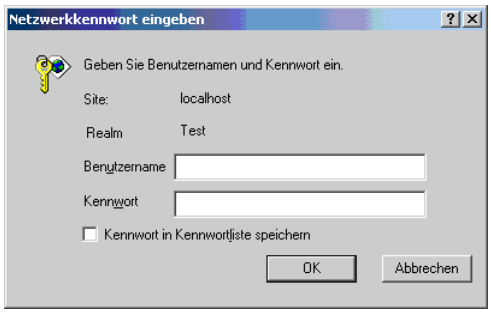


Bild 3-67 Anmeldung des Internet Explorers

Site bezeichnet die Adresse des Servers (hier localhost). Realm ist der Sicherheitsbereich und entspricht dem Attribut `REALM`.

Das REALM-Attribut kann dafür verwendet werden, ein Login des Users zu ermöglichen / zu erzwingen.

Hinweis

Für ein Verzeichnis kann nur ein REALM angegeben werden. In einer Verzeichnishierarchie dürfen sich unterschiedliche REALMS nicht überlagern, sie müssen getrennt sein.

Da es sich bei dem Zugriff auf die Datei-Objekte um einen hierarchischen Zugriff handelt, ist es denkbar, dass verschiedene Hierarchieebenen verschiedene Sicherheitsgruppen haben.

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE>
      <Motion REALM="Bediener">
         <Tests REALM="Tester" >
            [...]
         </Tests >
      </Motion>
   </BASE>
   [...]
</SERVERPAGES>
```

Ein Zugriff auf Tests und dessen Inhalt ist nicht möglich. Selbst wenn ein User in den Sicherheitsgruppen "Bediener" und "Tester" ist.

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE>
      <Motion SECUREGROUP="Bediener">
         [...]
      </Motion>
      <Tests SECUREGROUP="Tester" >
         [...]
      </Tests >
   </BASE>
[...]
</SERVERPAGES>
```

Hier hat ein User mit den Sicherheitsgruppen "Bediener" und "Tester" Zugriff auf Motion und Tests sowie ihre untergeordneten Objekte.

Siehe auch

<USERDATABASE> (Seite 281)

Attribut REALM (Seite 285)

3.4.3.8 READ

Leseberechtigung mit dem READ Attribut einrichten

Ist für ein Verzeichnis das `READ`-Attribut angegeben, muss der User Mitglied einer der beim `READ`-Attribut angegebenen Gruppen sein. Bei `READ` können mehrere Gruppen angegeben werden, diese müssen mit Komma getrennt sein, es dürfen keine Whitespace-Zeichen verwendet werden.

Beispiel

```
<MyDir READ="User,Administrator" />
```

User die den Gruppen User oder Administrator (oder beiden) angehören, dürfen den Inhalt des Verzeichnisses lesen.

Hat ein User kein Leserecht, d. h. er gehört keiner der bei `READ` angegeben Gruppen an, so wird eine FORBIDDEN Meldung erzeugt. Es wird kein Login beim Client initiiert.

Ist bei einem Verzeichnis kein `READ`-Attribut vorhanden, so ist der Lesezugriff grundsätzlich erlaubt.

Siehe auch

Attribut READ (Seite 284)

3.4.3.9 WRITE

Schreibberechtigungen mit dem WRITE-Attribut setzen

Besitzt ein Verzeichnis ein `WRITE`-Attribut, und ist der eingeloggte User Mitglied einer der angegebenen Gruppen, so darf der User in diesem Verzeichnis nur neue Dateien anlegen.

Er darf:

- keine neuen Verzeichnisse anlegen
- keine Dateien überschreiben
- keine Dateien löschen
- neue Dateien anlegen

Hinweis

Zum Anlegen von Dateien muss der User auch READ-Rechte haben!

Siehe auch

Attribut WRITE (Seite 286)

3.4.3.10 MODIFY

Verzeichnisse für die Modifikation freischalten

Besitzt ein Verzeichnis ein MODIFY-Attribut, und ist der eingeloggte User Mitglied einer der angegebenen Gruppen, so darf der User in diesem Verzeichnis alle Schreiboperationen ausführen:

Er darf:

- neue Verzeichnisse anlegen
- Dateien überschreiben
- Dateien löschen
- neue Dateien anlegen

Der User muss auf dem Verzeichnis natürlich auch READ-Rechte haben, sonst hätte er erst keinen Zugriff auf das Verzeichnis.

Siehe auch

Attribut MODIFY (Seite 283)

3.4.3.11 Anlegen von Verzeichnissen und Dateien

Werden Verzeichnisse oder Dateien angelegt, so erben sie die Berechtigungen des Verzeichnisses, in dem Sie liegen.

Rechte können durch den Verzeichnisbrowser nicht geändert werden, sondern nur direkt durch die Änderung der WebCfg.xml Datei.

3.4.3.12 Browsen des Dateisystems

Der Webserver bietet die Möglichkeit ein (physikalisches) Verzeichnis im Client zu visualisieren.

Hierzu muss das Attribut BROWSEABLE bei dem LOCALLINK-Tag oder das globale <BROWSEABLE>-Tag auf TRUE gesetzt werden.

Spricht ein Client diesen Link an, wird eine Verzeichnisansicht des Verzeichnisses erzeugt. Von diesem Verzeichnis aus kann auch in Unterverzeichnisse navigiert werden (oder auch in weiter oben gelegene Verzeichnisse, wenn für diese das Browsen auch erlaubt ist).

Man kann, ausreichend Rechte vorausgesetzt, Dateien senden, empfangen und löschen, sowie Verzeichnisse anlegen und löschen. Das Aussehen des Verzeichnisses im Client ist frei konfigurierbar.

Das <DEFAPP>-Tag = "Default Applikation" dient zum Konfigurieren des Verzeichnis Browsers.

Wenn kein Authentifizierungsmechanismus im Webserver vorhanden ist, sind verändernde Zugriffe generell nicht gestattet (siehe Sicherheitskonzept).

```
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
   [...]
   <BASE>
      <www LOCALLINK="/UserData" BROWSEABLE="TRUE"
           REALM="Bediener"/>
      <Test.mcs LINK="/Tests/Test.mcs/"/>
      <Default.mcs>
         <![CDATA[
         <HTML>
            <HEAD>
         [...]
         ]]>
      </Default.mcs>
   </BASE>
   [...]
</SERVERPAGES>
```

In diesem Beispiel würde eine Verzeichnisansicht des lokalen Verzeichnisses "/UserData" (Relativ zu WWWRoot!) an den Client zurückgegeben, wenn dieser die URL /www anfordert und sich als ein Benutzer der REALM "Bediener" authentifiziert hat.

Verändernde Zugriffe auf das Verzeichnis sind nicht möglich, da für den Verzeichniseintrag kein `WRITE` oder `MODIFY`-Attribut angegeben wurde.

3.4.3.13 Datei-Zugriff über FTP

Einleitung

Die Funktion "Datei-Zugriff über FTP" ermöglicht es, gezielt auf Dateien des Speicherkärtchens zuzugreifen. Ein FTP-Client kann z. B. ein Windows Explorer sein.

Bedingung

In der Datei WebCfg.xml muss sich ein Anwender in der Gruppe "FTPUser" befinden, um sich bei FTP anmelden zu können.

Die Datei WebCfg.xml wir beim ersten Hochlauf erzeugt.

Ausschnitt aus der WebCfg.xml

Im nachfolgend abgebildeten Ausschnitt aus der WebCfgxml wurde die Gruppe "FTPUser" dem User "simotion" zugeordnet.

```
<UserDataBase>
    <FILE NAME="UserDataBase.xml">
        <![CDATA[
            <?xml version="1.0" encoding="UTF-8"?>
            <UserDataBase>
                        <USER NAME="anonymous" PASSWORD="anonymous">
                                    <DESCRIPTION>Anonymous</DESCRIPTION>
                                    <GROUP NAME="Anyone"/>
                                    <GROUP NAME="OPC_XML"/>
                        </USER>
                        <USER NAME="internal" PASSWORD="internal">
                                    <DESCRIPTION>Internal user</DESCRIPTION>
                                    <GROUP NAME="Anyone"/>
                        </USER>
                        <USER NAME="simotion" PASSWORD="simotion">
                                    <DESCRIPTION>Default User</DESCRIPTION>
                                    <GROUP NAME="Administrator"/>
                                    <GROUP NAME="FTPUser"/>
                                    <GROUP NAME="Anyone"/>
                                    <GROUP NAME="OPC_XML"/>
                        </USER>
            </UserDataBase>
        ]]>
    </FILE>
</UserDataBase>
```

⚠ WARNUNG
Vorsicht beim Zugriff auf Systemdateien.

Hinweis

Der FTP-Zugriff wird nur im Binär-Modus unterstützt.

3.5 Anwenderdefinierte Seiten

3.5.1 Einleitung

Individuell gestaltete Webseiten mit Zugriff auf die Gerätedaten

SIMOTION IT DIAG bietet die Möglichkeit, individuell gestaltete Webseiten zu erstellen. Die Möglichkeiten der Gestaltung dieser Webseiten werden in diesem Abschnitt beschrieben.

Für den Zugriff auf Gerätedaten stehen zwei Varianten zur Auswahl::

- die JavaScript Bibliotheken opcxml.js und appl.js
- die MiniWeb Server Language (MWSL)

Das folgende Bild zeigt ein Beispiel einer selbst gestalteten Seite, die Variablen in Tabellenform anzeigt.

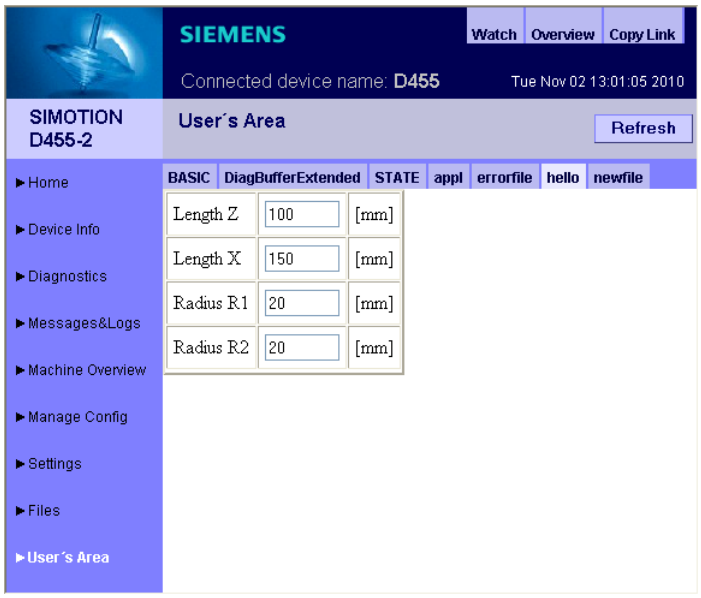


Bild 3-68 Anwenderdefinierte Seite

3.5.2 Konvertierung von Standard HTML-Seiten in Binärdateien

IT DIAG verwendet intern einen eigenen Dateityp ".mbs" zur Ausgabe von HTML-Seiten. Mit den Konvertierungstools U7NW2XMX.exe (Grafikoberfläche) oder U7NW2XCX.exe (Kommandozeile) ist es ab der SIMOTION Steuerung V4.1 möglich, diese Dateien zu erstellen.

Für die Konvertierung müssen MCS-Dateien erstellt werden, die den gleichen Grundaufbau wie Standard HTML-Seiten haben. Durch zusätzliche IT DIAG spezifische Tags und Funktionen kann auf Gerätedaten zugegriffen und die Darstellung der Seiten optimiert werden.

Ablauf der HTML-Seiten Erstellung und Konvertierung

1. Entwurf der HTML-Seiten mit einem Tool freier Wahl. Die Seiten erhalten die Dateiendung ".mcs".
2. Das Konvertierungstool mit dem Quellverzeichnis und Zielverzeichnis aufrufen. Dabei entstehen im Zielverzeichnis die konvertierten Dateien mit demselben Namen wie die zugehörigen Quelldateien und der Dateiendung ".mbs".
3. Die Dateien aus dem Zielverzeichnis auf die Speicherkarte der Steuerung in das Verzeichnis \USER\SIMOTION\HMI\FILES kopieren. Alternativ können Dateien über die Files Seite an das Gerät übertragen werden.

Umwandlung in Binärdateien mit dem Konvertierungstool

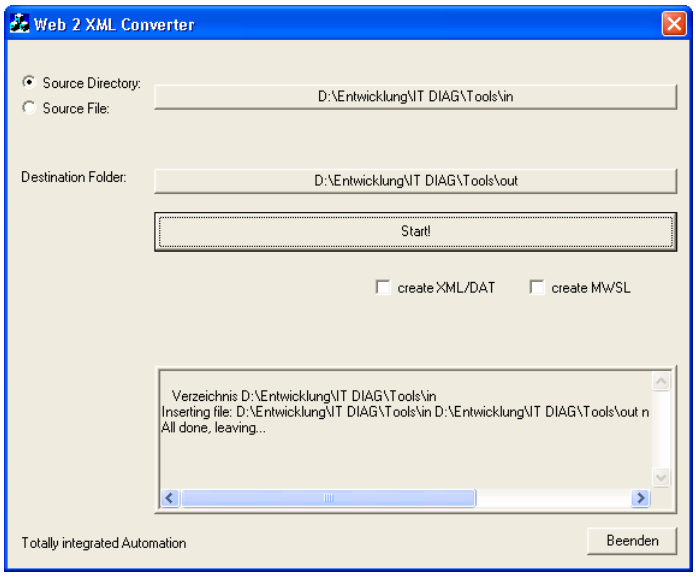


Bild 3-69 Konvertierungstool U7NW2XCX.exe

Optional können Sie eine Quelldatei oder einen Quellpfad wählen. Bei Angabe eines Quellpfades werden alle *.mcs-Dateien ab diesem Pfad konvertiert.

Durch drücken des **Start**-Buttons werden die HTML-Seiten konvertiert und im Zielpfad abgelegt.

Umwandlung in Binärdateien mit dem Kommandozeilen-Konvertierungstool

Die Aufrufsyntax lautet:

```
U7NW2XCX.exe [-f] <Quelle> <Ziel>
```

Aufrufparameter:

-f	Optional: Verarbeitung einer einzelnen Datei
Quelle	Quellpfad oder Quelldatei. Bei Angabe des Quellpfades werden alle *.mcs-Dateien ab diesem Pfad konvertiert.
Ziel	Zielpfad

Hinweis

Die Konvertierungstools finden Sie auf der SIMOTION SCOUT AddOn DVD im Verzeichnis 4_Accessories\Simotion_IT\6_Tools.

Beispiel einer anwenderdefinierten Seite

Die Erstellung einer HTML-Seite kann mit einem beliebigen Texteditor erfolgen.

```
<html>
  <head/>
  <body>
    <br>  State: <MWSL><!--WriteVar("DeviceInfo.BZU");--></MWSL>
  </body>
</html>
```

Newfile.mcs

Der Quelltext wird in diesem Beispiel als Datei mit dem Namen Newfile.mcs abgespeichert.

Zur Ausgabe von Gerätedaten wird in diesem Beispiel die MWSL benutzt. Aufbau einer MWSL Datei (Seite 163)

In den MWSL-Ausdrücken wird mittels des Variablen Provider auf die Gerätedaten zugegriffen. Gruppe DeviceInfo (Seite 249)

Nach der Konvertierung erhält man die Datei Newfile.mbs, die man auf das Gerät über die Files Seite kopiert.

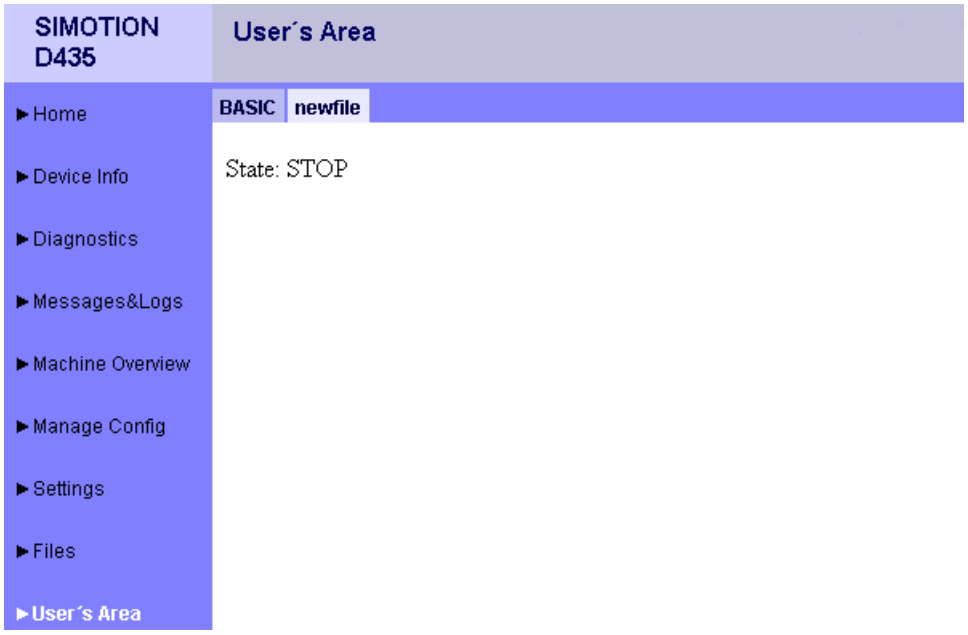


Bild 3-70 Newfile.mbs im Browser

Voraussetzung für die Anzeige der Seite ist die korrekte Einstellung der User's Area. In diesem Beispiel wurde die EmbeddedSimple Variante gewählt. Eingebettete anwenderdefinierte Seiten (Seite 126)

3.5.3 Anwenderdefinierte Startseite

Anstelle der Startseite der Standard-Diagnoseseiten der Steuerung können Sie eine eigene Startseite erstellen und gegen die Standardseiten austauschen. Hierfür müssen Sie in der Datei WebCfg.xml die Default-Seite des Webservers umstellen.

Vorgehensweise

1. Erstellen Sie eine eigene Startseite und speichern Sie diese z. B. unter dem Namen `MYINDEX.MCS` ab.
2. Übertragen Sie die Startseite über die Seite **Files** auf die Speicherkarte des SIMOTION Geräts.
3. Öffnen Sie die Datei WebCfg.xml mit einem Ihnen verfügbaren Editor. Sie finden die Datei auf der Liefer-DVD im Verzeichnis 3_Configuration im Default-Zustand oder in ggf. modifizierten Zustand auf der Speicherkarte des SIMOTION Geräts im Verzeichnis \USER\SIMOTION\HMICFG.
4. Ersetzen Sie den Dateinamen index.mcs in den `<SERVEROPTIONS>` im Element `<DEFAULTDOCUMENT VALUE="index.mcs" />` durch den Namen Ihrer Startseite inkl. des Pfadnamens "files" (im Verzeichnis FILES werden alle anwenderdefinierten HTML-Seiten abgelegt).
5. Speichern Sie die geänderte WebCfg.xml über die Seite "Settings" auf der Speicherkarte ab.

Ein Beispiel für das Einbetten einer HTML-Seite befindet sich im Kapitel 'Virtuelles Dateisystem' (Seite 108).

3.5.4 Eingebettete anwenderdefinierte Seiten

Einbindung Anwenderdefinierter Seiten

Ab Version 4.1.3 ist es möglich, anwenderdefinierte Seiten ín das Framework der SIMOTION Standardseiten einzubetten.

Bild 3-71 User's Area mit eingebetteter Seite

Das Menu der **User`s Area** bindet die Dateien des Ordner FILES auf zwei verschiedene Arten ein:

- EmbeddedSimple: Die Seite **User`s Area** lädt alle im Ordner FILES enthaltenen Webseiten als Reiter. Angezeigt wird der Dateiname ohne Dateiendung.
- Embedded: Die Seite lädt einen vom Anwender frei definierbaren Reiterleiste.

Zwischen EmbeddedSimple und Embedded kann auf der Settings (Seite 86) Seite umgeschaltet werden.

Einstellungen in der WebCfg.xml

In der WebCfg.xml kann das Aussehen der User's Area durch die Konfigurationskonstanten <UserArea> und <UserDir> eingestellt werden.

Mit dem Tag <UserArea> kann die Art der Reiterdarstellung eingestellt werden. (Der Defaultfall ist fett dargestellt):

```
<UserArea>( StandAlone | Embedded | EmbeddedSimple )</UserArea>
```

<UserDir> bezeichnet das Verzeichnis für die Reiterdateien relativ zum FILE-Verzeichnis.

```
<UserDir></UserDir>
```

StandAlone

```
<UserArea>StandAlone</UserArea>
```

Der Zugriff auf die **User's Area** ist fest mit der Datei user.mcs verknüpft. Voraussetzung für die Anzeige, der **User's Area** ist, das Vorhandensein und die Abrufbarkeit dieser Datei.

Automatische

```
<UserArea>EmbeddedSimple</UserArea>
```

Bei dieser Option werden alle im durch <UserDir> bezeichneten Verzeichnis gefundenen Dateien zur Bildung des Reiters verwendet.

Als Titel wird der jeweilige Dateiname ohne Namenserweiterung verwendet. Der entsprechende Link verweist auf diese Datei.

Benutzung des Menü-Editors

```
<UserArea>Embedded</UserArea>
```

Wenn diese Option gesetzt ist, wird in der **User's Area** der Menü-Editor (Seite 128) angezeigt, mit dem sich Menüs individuell gestalten lassen.

Beispiel WebCfg.xml:

```
<SERVERPAGES version="59.00">
  [...]
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <UserArea>Embedded</UserArea>
      <UserDir/>
    </USERCONFIG>
  </CONFIGURATION_DATA>
  [...]
</SERVERPAGES>
```

3.5.5 Menü-Editor

Individuelle Menüs mit dem Menü-Editor erstellen

Über den Link **Menu editor** kann ab Version 4.1.3 der Menü-Editor aufgerufen werden, der eine individuelle Gestaltung des Menüs der User Area gestattet.

Voraussetzungen zur Benutzung des Menü-Editors

Die Benutzung des Menü-Editors setzt voraus, dass in der WebCfg.xml bei den Konfigurationsdaten die `<UserArea>` auf `Embedded` gesetzt wird.

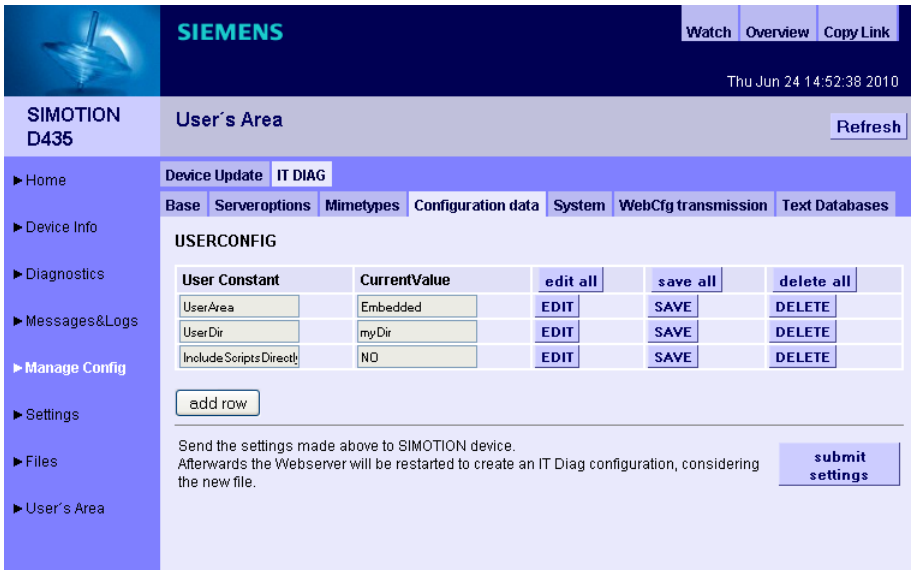


Bild 3-72 Menü Editor Configuration Data

Danach muss auf der **Settings** Seite die Option **Enable user menu editor** angewählt werden.

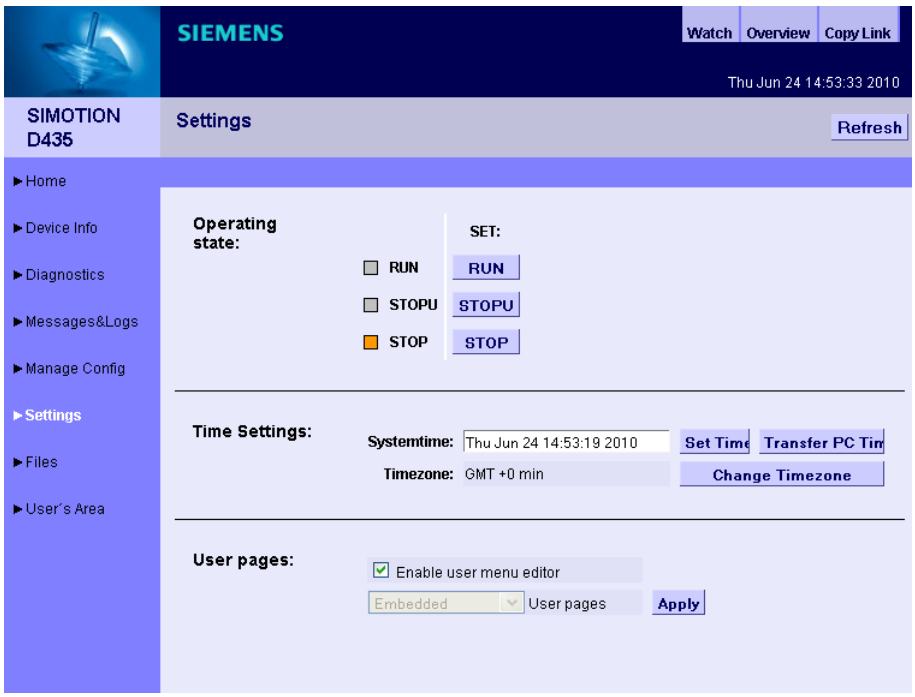


Bild 3-73 Menü Editor Settings

Arbeiten mit dem Menü-Editor

Die **User's Area** Seite enthält dann den Reiter **Menu editor**.

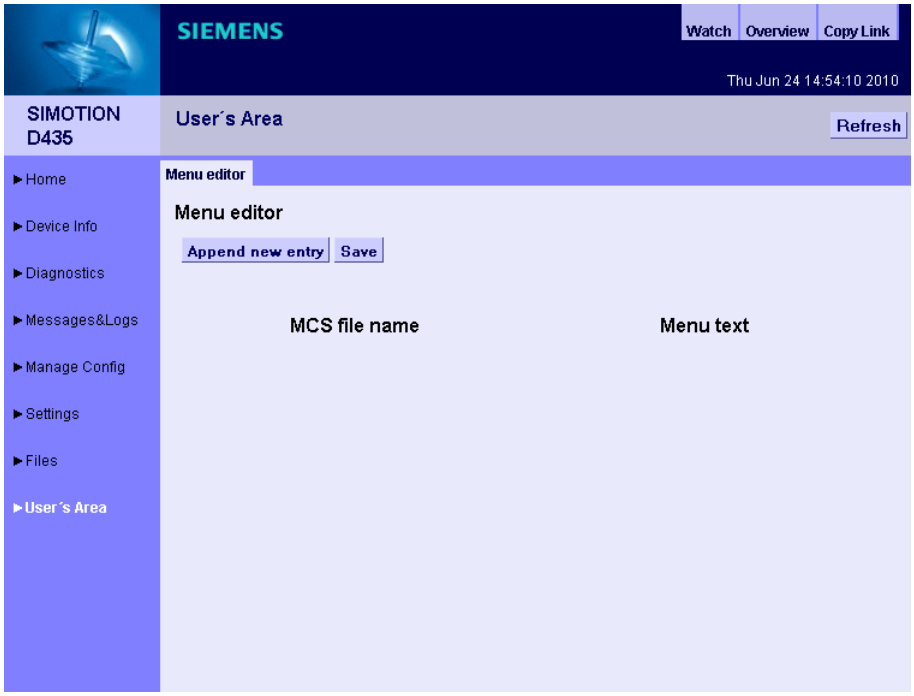


Bild 3-74 Erster Start des Menü-Editors

Nach dem ersten Start des Menü-Editors präsentiert sich eine weitgehend leere Seite.

Über den Button **Append new entry** können neue Menüeinträge erstellt werden.

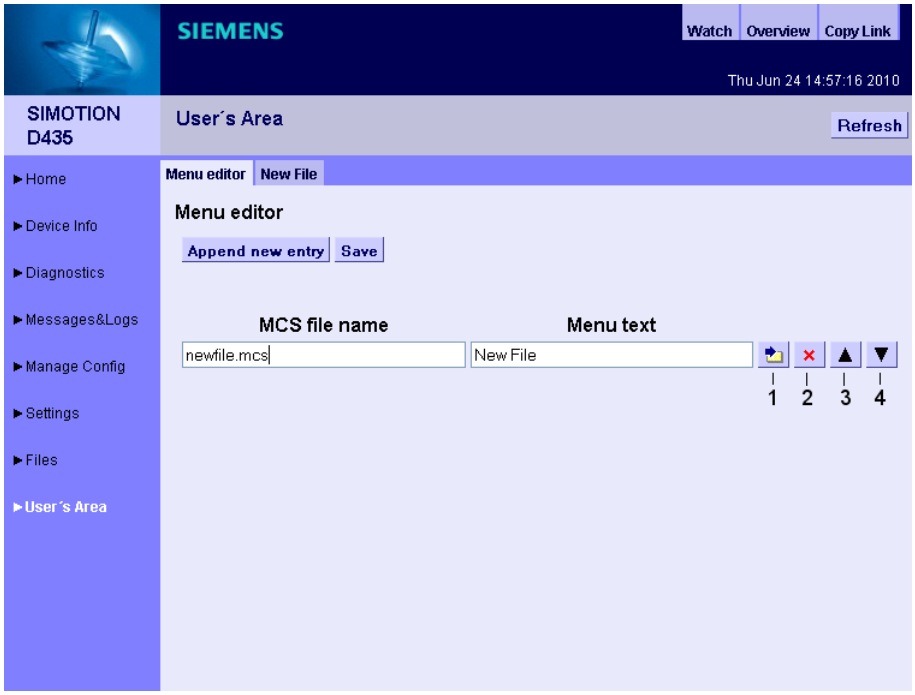


Bild 3-75 Menü-Editor mit mehreren Einträgen

Im obigen Bild wurde die Datei newfile.mcs hinzugefügt. Über die Buttons können Dateien hinzugefügt, gelöscht oder ihre Position verschoben werden.

In der Spalte **MCS file name** werden die Dateinamen der Dateien, die zu dem entsprechenden Menüpunkt angezeigt werden sollen, eingetragen.

Die Spalte **Menu text** enthält den Titel des Menüeintrags.

Der Button ① dient zur Anlage neuer Menüeinträge, die vor dem aktuellen eingefügt werden.

Der Button ② löscht den entsprechenden Menüeintrag.

Der Button ③ verschiebt den Menüeintrag nach oben.

Der Button ④ verschiebt den Menüeintrag nach unten.

3.5.6 JavaScript und Webservices

3.5.6.1 Variablenzugriff mit JavaScript und Webservices

Zugriff auf ein Gerät mit der JavaScript-Bibliothek

Mithilfe der DOM-Funktionalität von JavaScript ist es möglich, einfache Webservice-Clients zu implementieren. Damit eröffnen sich zahlreiche neue Möglichkeiten innerhalb eines Browsers, z. B.:

- Lesen und zyklisches Aktualisieren von Variableninhalten mithilfe eines OPCXML-Read-Kommandos
- Schreiben von Variablen mithilfe eines OPC XML-DA Write-Kommandos
- Browsen des gesamten SIMOTION-Variablenhaushalts
- Einrichten und Abfragen einer OPC XML-DA Subscription

Die Funktionalität wird von mehreren JavaScript Files zur Verfügung gestellt:

- opcxml.js: beinhaltet Funktionen für den Aufbau der benötigten XML-Dokumente und die Kommunikation mit einem OPC XML-DA Server
- appl.js: setzt auf opcxml.js auf und implementiert die folgenden Objekte:
  - Variablenbowser: Darstellung des SIMOTION-Variablenhaushalts in einer Baumstruktur innerhalb des Browsers
  - Property Viewer: Darstellung von Variableneigenschaften (Wert, Datentyp, Zugriffsrechte, Enums) in Form einer Tabelle innerhalb eines Browsers. Die Tabelle beinhaltet bei schreibbaren Variablen ein Eingabefeld zum Verändern des Variableninhalts.
  - Watchtabelle: Darstellung einer Watchtabelle im Browser

3.5.6.2 Kommunikation mit dem OPC XML-DA Server (opcxml.js)

OPCReadRequest

Mithilfe der Klasse `OPCReadRequest` können die Werte für eine Liste von Variablen gelesen werden.

```
function OPCReadRequest(parLocaleId,parResultCB)
```

Übergabeparameter:

- `parLocaleId`: Sprachkennung ("DE", "EN")
- `parResultCB`: Callback-Funktion, die vom Aufrufer bereitgestellt werden muss.
  Diese Funktion wird von OPCReadRequest aufgerufen, wenn eine Antwort vom OPC XML-DA Server eingetroffen ist. Das OPCReadRequest-Objekt wird automatisch entsorgt (Aufruf der Methode "destructor"), wenn die Callback-Funktion "true" als Rückgabewert liefert.

  ```
  function OPCReadRequestCB(parResponse)
  ```

  Übergabeparameter:

  – `parResponse`: Array von ItemValues mit dem Ergebnis des Leseauftrags.

  ```
  function ItemValue()
  {
     this.mItemPath;
     this.mItemName;
     this.mItemHandle;
     this.mItemValue;
     this.mItemResultId;
  }
  ```

Ist `mItemResultId` definiert, so ist ein Fehler beim Lesen aufgetreten. In diesem Fall enthält `mItemResultId` die OPC XML-DA Fehler-ID.

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
         function read()
         {
             var tmpReadCB = function(parResponse)
             {
                 tmpResultStr = "";
                 for (var tmpIndex = 0; tmpIndex < parResponse.length;
                      tmpIndex++)
                 {
                    var tmpItemValue = parResponse[tmpIndex];
                    var tmpValue = (tmpItemValue.mItemValue) ?
                                   tmpItemValue.mItemValue :
                                   tmpItemValue.mItemResultId;
                    tmpResultStr += tmpItemValue.mItemPath
                                   + "::"
                                   + tmpItemValue.mItemName
                                   + " = "
                                   + tmpValue
                                   + "\n";
                 }
                 alert(tmpResultStr);
                 return true;
              }
              var tmpReadRequest = new OPCReadRequest("DE",tmpReadCB);
              tmpReadRequest.addItem("SIMOTION","var/userdata.user1");
              tmpReadRequest.addItem("SIMOTION","var/userdata.user2");
              tmpReadRequest.addItem("SIMOTION","var/userdata.user10");
              tmpReadRequest.sendReadRequest();
         }
      </script>
      <title>Insert title here</title>
   </head>
   <body>
      <input type="button" onclick="read();" value="Read"/>
   </body>
</html>
```

OPCGetPropertiesRequest

Mithilfe der Klasse `OPCGetPropertiesRequest` können die Eigenschaften von Variablen gelesen werden:

- Werte
- Datentypen
- Zugriffsrechte
- Enum-Komponenten

```
function OPCGetPropertiesRequest(parLocaleId,parResultCB)
```

Übergabeparameter:

- `parLocaleId`: Sprachkennung ("DE", "EN")
- `parResultCB`: Callback-Funktion, die vom Aufrufer bereitgestellt werden muss
  Diese Funktion wird von OPCGetPropertiesRequest aufgerufen, wenn eine Antwort vom OPC XML-DA Server eingetroffen ist. Das OPCGetPropertiesRequest-Objekt wird automatisch entsorgt (Aufruf der Methode "destructor"), wenn die Callback-Funktion "true" als Rückgabewert liefert.

  ```
  function OPCGetPropertiesRequestCB(parResponse)
  ```
  parResponse: Array von PropertyResults, die die Properties der Variablen beinhalten:

  ```
  function PropertyResult()
  {
     this.mItemPath;
     this.mItemName;
     this.mResultId;
     this.mValue;
     this.mType;
     this.mAccessRights;
     this.mEnums;
  }
  ```

Anwenderschnittstelle:

- `addItem(parItemPath,parItemName)` fügt eine Variable zur Variablenliste hinzu
- `removeItem(parItemHandle)` löscht eine Variable aus der Variablenliste
- `sendGetPropertiesRequest()` schickt den Leseauftrag ab
- `destructor()` gibt das gesamte Auftragsobjekt frei

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <title>GetProperties</title>
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
      function getProperties()
      {
         var tmpGetPropertiesCB = function(parResponse)
         {
             tmpResultStr = "";
             for (var tmpIndex = 0; tmpIndex < parResponse.length;
                   tmpIndex++)
             {
                 var tmpPropertyResult = parResponse[tmpIndex];
                 var tmpEnums = "";
                 if (tmpPropertyResult.mEnums &&
                     (tmpPropertyResult.mEnums.length > 0))
                {
                    for (var tmpIndex = 0;
                         tmpIndex < tmpPropertyResult.mEnums.length;
                         tmpIndex++)
                    {
                        tmpEnums += " " +
                           tmpPropertyResult.mEnums[tmpIndex] + "\n";
                    }
                 }
                 if (!tmpPropertyResult.mResultId)
                 {
                   tmpResultStr += tmpPropertyResult.mItemPath +
                                   "::" +
                                   tmpPropertyResult.mItemName +
                                   ":\n Type = " +
                                   tmpPropertyResult.mType +
                                   "\n value = " +
                                   tmpPropertyResult.mValue +
                                   "\n AccessRights = " +
                                   tmpPropertyResult.mAccessRights +
                                   "\n Enums = \n" + tmpEnums +"\n";
                 }
                 else
                 {
                     tmpResultStr += tmpPropertyResult.mItemPath +
                                     "::" + tmpPropertyResult.mItemName
                                     + ":\n ResultId = " +
                                     tmpPropertyResult.mResultId +
                                     "\n\n";
```

```
                    }
                }
                alert(tmpResultStr);

                return true;
            }
            var tmpGetPropertiesRequest =
             new OPCGetPropertiesRequest("DE",tmpGetPropertiesCB);
            tmpGetPropertiesRequest.addItem("SIMOTION",
                                            "var/userdata.user1");
            tmpGetPropertiesRequest.addItem("SIMOTION",
                                            "var/userdata.user20");
            tmpGetPropertiesRequest.addItem("SIMOTION",
                                            "dev/Service.BZU.value");
            tmpGetPropertiesRequest.sendGetPropertiesRequest();
          }
      </script>
   </head>
   <body>
      <input type="button" onclick="getProperties();"
             value="getProperties"/>
   </body>
</html>
```

OPCWriteRequest

`OPCWriteRequest` schreibt drei Werte einer oder mehrerer Variablen.

```
function OPCWriteRequest(parLocaleId,parResultCB)
```

Übergabeparameter:

- `parLocaleId`: Sprachkennung ("DE", "EN")
- `parResultCB`: Callback-Funktion, die vom Aufrufer bereitgestellt werden muss
  Die Funktion wird nach Abschluss des Sendeauftrags aufgerufen.

  ```
  function OPCWriteRequestCB(parResultList)
  ```
  parResultList ist ein Array von ItemValues, das die Ergebnisse des Schreibauftrags beinhaltet.

  ```
  function ItemValue()
  {
     mItemPath
     mItemName
     mItemHandle
     mItemValue
     mItemResultId
  }
  ```
  Das OPCWriteRequest-Objekt wird automatisch entsorgt (Aufruf der Methode "destructor"), wenn die Callback-Funktion "true" als Rückgabewert liefert.

Anwenderschnittstelle:

- `addItem(parItemPath,parItemName,parType)` fügt eine Variable zur Variablenliste hinzu, liefert ein Variablenhandle zurück, mit dem die Variable innerhalb des Auftrags referenziert werden kann. ParType kennzeichnet den OPC XML-DA Datentyp, mit dem geschrieben werden soll. Wird parType nicht übergeben, so wird der Datentyp "xsi::string" angenommen.
- `removeItem(parItemHandle)` entfernt eine Variable aus der Variablenliste
- `setItemValue(parItemHandle,parValue)` setzt für eine Variable den Wert, der geschrieben werden soll.
- `sendWriteRequest()` schickt den Schreibauftrag ab
- `destructor()` gibt alle vom Schreibobjekt belegten Ressourcen frei

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
      function writeValues()
      {
          var tmpWriteCB = function(parWriteResult)
          {
              var tmpString = "";
              for (var tmpIndex = 0; tmpIndex < parWriteResult.length;
                   tmpIndex++)
              {
                var tmpItemValue = parWriteResult[tmpIndex];
                var tmpValue = (tmpItemValue.mItemResultId) ?
                 tmpItemValue.mItemResultId : tmpItemValue.mItemValue;
                tmpString += tmpItemValue.mItemPath + "::" +
                  tmpItemValue.mItemName + " = " + tmpValue + "\n";
              }
              alert(tmpString);
          }
          var tmpWrite = new OPCWriteRequest("DE",tmpWriteCB);
          var tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                               "var/userdata.user1");
          tmpWrite.setItemValue(tmpItemHandle,"123");
          tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                           "var/userdata.user2");
          tmpWrite.setItemValue(tmpItemHandle,"234");
          tmpItemHandle = tmpWrite.addItem("SIMOTION",
                                           "var/userdata.user10");
          tmpWrite.setItemValue(tmpItemHandle,"345");
          tmpWrite.sendWriteRequest();
      }
      </script>
      <title>Write</title>
   </head>
<body>
<input type="button" value="Write" onclick="writeValues()"/>
</body>
</html>
```

OPCBrowseRequest

Mithilfe der Klasse `OPCBrowseRequest` kann der Variablenhaushalt einer Steuerung durchsucht werden.

```
function OPCBrowseRequest(parLocaleId,parResultCB)
```

Übergabeparameter:

- `parLocaleId`: Sprachkennung ("DE", "EN")
- `parResultCB`: Callback-Funktion, die vom Aufrufer bereitgestellt werden muss. Die Funktion wird nach Abschluss des Sendeauftrags aufgerufen.

  ```
  function OPCBrowseRequestCB(parResult,parItemPath,parItemName)
  ```

  `parResult` ist ein Array vom Typ `BrowseResult` und enthält die Browse-Informationen.

  ```
  function BrowseResult()
  {
     mItemPath;
     mItemName;
     mName;
     mIsItem;
     mHasChildren;
  }
  ```

  `parItemPath` und `parItemName` Pfad und Name desjenigen Verzeichnisses, dessen Inhalt in `BrowseResult` enthalten ist.

Anwenderschnittstelle:

- `sendBrowseRequest(parItemPath,parItemName)` schickt den Browseauftrag ab `parItemPath` und `parItemName` sind Pfad und Name des Directories, das gebrowst werden soll.

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
       <meta http-equiv="Content-Type" content="text/html;
             charset=ISO-8859-1">
       <script type="text/javascript" src="/common.js"></script>
       <script type="text/javascript" src="/opcxml.js"></script>
       <script type="text/javascript">
       function browse()
       {
           var tmpBrowseRequestCB = function(parBrowseResult,
                                             parItemPath,
                                             parItemName)
           {
             var tmpString = parItemPath + "::" + parItemName + "\n";
             for (var tmpIndex = 0; tmpIndex < parBrowseResult.length;
                  tmpIndex++)
             {
                 var tmpBrowseResult = parBrowseResult[tmpIndex];
                 tmpString += tmpBrowseResult.mItemName + "\n";
             }
             alert(tmpString);
           }
           var tmpBrowseRequest =
             new OPCBrowseRequest("DE",tmpBrowseRequestCB);
           tmpBrowseRequest.sendBrowseRequest("SIMOTION","var/");
        }
        </script>
        <title>Browse</title>
   </head>
   <body>
       <input type="button" value="Browse" onclick="browse();"/>
   </body>
</html>
```

OPCSubscriptionRequest

Mithilfe der Klasse `OPCSubscriptionRequest` kann eine OPC XML-DA Subscription eingerichtet, gepollt und gelöscht werden.

```
function OPCSubscriptionRequest(parLocaleId,parResultCB,parCancelCB)
```

Übergabeparameter:

- `parLocaleId` Sprachkennung ("DE", "EN")
- `parResultCB` Callbackfunktion, die vom Anwender zur Verfügung gestellt werden muss. Sie wird nach dem Einrichten und nach einem Refresh aufgerufen.

  `function OPCSubscriptionRequestCB(parResultList,parResult)`
  `parResultList` ist ein Array vom Typ `OPCItemValue`, das die ermittelten Variablenwerte beinhaltet.

  ```
  function OPCItemValue()
  {
     mItemPath
     mItemName
     mItemHandle
     mItemValue
     mItemResultId
  }
  ```

- `parCancelCB` Callbackfunktion, die vom Anwender zur Verfügung gestellt werden muss. Sie wird nach dem Freigeben einer Subscription aufgerufen.

  `function OPCSubscriptionCancelCB()`
  Die Funktion hat keine Übergabeparameter.

Anwenderschnittstelle:

- `addItem(parItemPath,parItemName)` fügt die übergebenen Variable zur internen Liste der Subscriptionvariablen hinzu
- `removeItem(parItemHandle)` löscht die übergebene Variable aus der Liste der Subscription-Variablen
- `cancel()` meldet eine active Subscription beim Server ab

- `refresh()` liest die aktuellen Variablenwerte
  Es werden nur Variablen übertragen, deren Werte sich seit der vorangehenden Abfrage geändert haben. Der erste Aufruf von refresh nach dem Erzeugen des OPCSubcription-Objektes bzw. der erste Aufruf nach einem cancel meldet die Subscription mit der aktuellen internen Variablenliste beim Server an. Refresh muss zyklisch aufgerufen werden. Der Holdtime-Mechanismus der OPC XML-DA Subscription wird nicht unterstützt, d. h., zwischen je 2 Refreshzyklen muss über einen JavaScript-Timer eine Wartezeit programmiert werden. Die Waittime der OPC XML-DA Subscription ist jedoch aktiv, d. h., eine Antwort auf einen Refreshaufruf erfolgt erst nach Ablauf der Waittime, wenn sich während der Laufzeit der Waittime kein Variablenwert ändert.
- `destructor()`
  Meldet die Subscription beim Server ab und gibt die vom Subscription-Objekt belegten Ressourcen frei. Destructor muss vor dem Freigeben des Objektes aufgerufen werden.

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
          var gloSubscription;
          var gloItemHandle_1;
          var gloItemHandle_2;
          function subscription()
          {
              if (!gloSubscription)
              {
                  var tmpSubscriptionCB = function(parValues)
                  {
                      for (var tmpIndex = 0;
                           tmpIndex < parValues.length;
                           tmpIndex++)
                      {
                          var tmpItemHandle =
                              parValues[tmpIndex].mItemHandle;
                          var tmpItemValue =
                              parValues[tmpIndex].mItemValue;
                          if (tmpItemHandle == gloItemHandle_1)
                          {
                              var tmpValueNode =
                                 document.getElementById("user1");
                              tmpValueNode.firstChild.nodeValue =
                                 tmpItemValue;
                          }
                          else if (tmpItemHandle == gloItemHandle_2)
                          {
                              var tmpValueNode =
                                 document.getElementById("user2");
                              tmpValueNode.firstChild.nodeValue =
                                  tmpItemValue;
                          }
                      }
                    var tmpTimerCB = function()
                    {
                        gloSubscription.refresh();
                    }
                    setTimeout(tmpTimerCB,300);
                }
                var tmpCancelCB = function()
                {
                    if (gloSubscription)
```

```
                        {
                            gloSubscription.destructor();
                            gloSubscription = null;
                        }
                    }
                    gloSubscription =
                       new OPCSubscriptionRequest("DE",
                            tmpSubscriptionCB,tmpCancelCB);
                    gloItemHandle_1 =
                       gloSubscription.addItem("SIMOTION",
                                               "var/userdata.user1");
                    gloItemHandle_2 =
                       gloSubscription.addItem("SIMOTION",
                                               "var/userdata.user2");
                    gloSubscription.refresh();
                 }
              }
              function cancel()
              {
                 if (gloSubscription)
                    gloSubscription.cancel();
              }
           </script>
              <title>Subscription</title>
           </head>
           <body>
              <div>
                 <input type="button" value="Start"
                        onclick="subscription();"/>
                 <input type="button" value="Cancel" onclick="cancel();"/>
              </div>
              <table>
                 <tr>
                     <td>user1</td>
                     <td id="user1">user1</td>
                 </tr>
                 <tr>
                     <td>user2</td>
                     <td id="user2">user2</td>
                 </tr>
              </table>
           </body>
</html>
```

OPCSubscriptionAutoRefresh

`OPCSubscriptionAutoRefresh` bietet die gleiche Funktionalität wie `OPCSubscriptionRequest`, allerdings erfolgt der timergesteuerte Aufruf der Refreshfunktion automatisch.

```
function OPCSubscriptionAutoRefresh(parLocaleId,parResultCB,parCancelCB,
 parCycleTime)
```

Übergabeparameter:

- `parLocaleId`
- `parResultCB`
- `parCancelCB`
  siehe OPCSubscriptionRequest
- `parCycleTime`
  Zykluszeit in ms, mit der die Refreshfunktion aufgerufen wird

Anwenderschnittstelle:

- `startRefresh()` startet den Abfragezyklus
- `cancel()` siehe OPCSubscriptionRequest
- `addItem(parItemPath,parItemName,parItemHandle)` siehe OPCSubscriptionRequest
  Nach dem Hinzufügen der Variablen wird der Refreshzyklus automatisch wieder aufgenommen.
- `removeItem(parItemHandle)`
- `destructor()`

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript">
         var gloSubscription;
         var gloItemHandle_1;
         var gloItemHandle_2;
         var gloItemHandle_3;
         function subscription()
         {
             if (!gloSubscription)
             {
                 var tmpSubscriptionCB = function(parValues)
                 {
                    for (var tmpIndex = 0; tmpIndex < parValues.length;
                         tmpIndex++)
                    {
                        var tmpItemHandle =
                            parValues[tmpIndex].mItemHandle;
                        var tmpItemValue =
                            parValues[tmpIndex].mItemValue;
                        if (tmpItemHandle == gloItemHandle_1)
                        {
                            var tmpValueNode =
                               document.getElementById("user1");
                            tmpValueNode.firstChild.nodeValue =
                               tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_2)
                        {
                            var tmpValueNode =
                                document.getElementById("user2");
                            tmpValueNode.firstChild.nodeValue =
                                tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_3)
                        {
                            var tmpValueNode =
                                document.getElementById("user3");
                            tmpValueNode.firstChild.nodeValue =
                                tmpItemValue;
                        }
                     }
                 }
                 var tmpCancelCB = function()
```

```
                {
                    if (gloSubscription)
                    {
                        gloSubscription.destructor();
                        gloSubscription = null;
                    }
                }
                gloSubscription =
                  new OPCSubscriptionAutoRefresh("DE",
                                tmpSubscriptionCB,tmpCancelCB,300);
                gloItemHandle_1 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user1");
                gloItemHandle_2 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user2");
                gloSubscription.startRefresh();
             }
          }
          function addVar()
          {
             if (gloSubscription)
             {
                 gloItemHandle_3 =
                    gloSubscription.addItem("SIMOTION",
                                            "var/userdata.user3");
             }
          }
          function removeVar()
          {
             if (gloSubscription)
             {
                 gloSubscription.removeItem(gloItemHandle_3);
             }
          }
          function cancel()
          {
              if (gloSubscription)
                  gloSubscription.cancel();
          }
   </script>
   <title>Auto refresh</title>
</head>
<body>
   <div>
       <input type="button" value="Start"
              onclick="subscription();"/>
       <input type="button" value="Add variable"
              onclick="addVar();"/>
       <input type="button" value="Remove variable"
              onclick="removeVar();"/>
       <input type="button" value="Cancel" onclick="cancel();"/>
```

```
        </div>
        <table>
           <tr>
              <td>user1</td>
              <td id="user1">user1</td>
          </tr>
          <tr>
              <td>user2</td>
              <td id="user2">user2</td>
         </tr>
         <tr>
            <td>user3</td>
            <td id="user3">user3</td>
         </tr>
        </table>
    </body>
</html>
```

3.5.6.3 Darstellung von OPC XML-DA Daten im Browser (appl.js)

In der JavaScript-Bibliothek **appl.js** sind Klassen für die Darstellung der mit OPC XML-DA Requests ermittelten Daten versammelt.

`ApplDataTable`

`ApplDataTable` implementiert eine dynamische Tabelle, in der Prozessvariablen dargestellt werden können. Die Variablenwerte werden mit einer OPC XML-DA Subscription zyklisch aktualisiert.

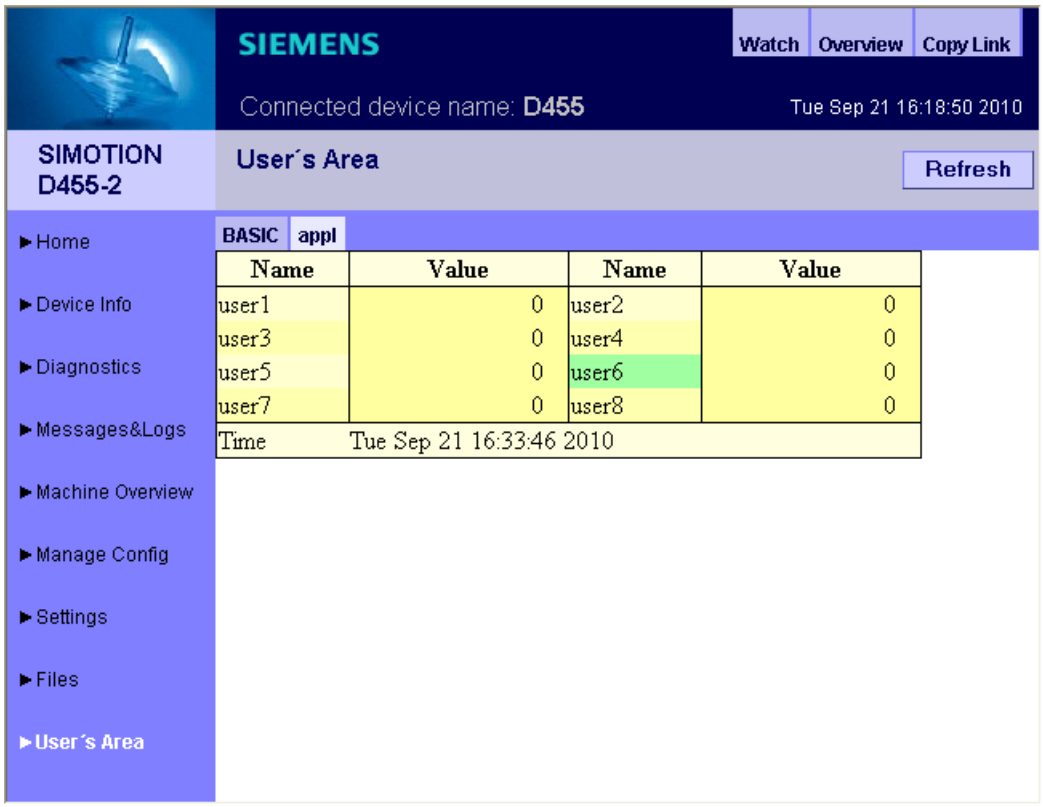


Bild 3-76 Beispiel der Implementierung einer dynamischen Tabelle (ApplDataTable)

```
function
ApplDataTable(parDocument,parClassName,parColumnClasses,parColumnIds
,parHeader)
```

Übergabeparameter:

- `parDocument` JavaScript-Dokument, das zum Erzeugen von Elementen benutzt wird
- `parClassName` wird im "Table"-Tag der HTML-Tabelle als "class"-Attribut eingetragen
- `parColumnClasses` Array, dessen "length"-Attribut die Anzahl der Tabellenspalten festlegt. Die Werte des Arrays werden als "class"-Attribut bei den Tabellenspalten (<td class="…">) verwendet.
- `parColumnIds`
  Array, dessen Werte zusammen mit dem später beschriebenen parRowId-Parameter für die "id"-Attribute der Tabellenspalten verwendet werden. Der Wert des "id"-Attributs entsteht durch das Aneinanderketten der ColumnId und der RowId (In dieser Reihenfolge).
- `parHeader` Array, das die Spaltenüberschriften der Tabelle enthält

Mithilfe der Übergabeparameter werden die "class"- und "id"-Attribute so gesetzt, dass unter Verwendung von Style Sheets die Darstellung der Tabelle eingestellt werden kann.

Anwenderschnittstelle:

- `addRow(parRowId,parRowClass)` hängt eine neue Zeile an die Tabelle an
  - `parRowId`: wird als Wert für das `id`-Attribut der Zeile (`<tr id="…">`) verwendet
  - `parRowClass`: wird als Wert für das "`class`"-Attribut der Zeile (<tr class="…">) verwendet
- `addElement(parElement,parDestructor,parColSpan,parColClass)`
  Fügt das mit `parElement` übergebene HTML-Element in die Tabelle ein.
  - `parElement` HTML-Element, das in die Tabelle eingefügt werden soll
  - `parDestructor` (optional)
    Funktion, die aufgerufen wird, wenn das HTML-Element aus der Tabelle gelöscht wird (braucht man im Internet-Explorer zur Vermeidung von Memeory Leaks).
  - `parColSpan` (optional)
    Gibt die Anzahl der Spalten an, über die sich das Element erstrecken soll.
  - `parColClass` (optional)
    Mit `parColClass` kann die beim Anlegen des `ApplDataTable`-Objektes vergebene ColClass überschrieben werden, falls für das aktuelle Element spezielle Formatierungen gemacht werden sollen.

- `addVariable(parPath,parName,parColSpan,parColClass)`
  Fügt eine Variable in die Tabelle ein. Der Wert der Variablen wird zyklisch aktualisiert.
  - `parPath`
    Variablenpfad (z. B. "SIMOTION" oder "SIMOTION diagnostics")
  - `parName`
    Variablenname (z. B. "var/userdata.user1")
  - `parColSpan` (optional)
    Siehe `addElement`
  - `parColClass` (optional)
    Siehe `addElement`
- `addText(parText,parColSpan,parColClass)` Fügt einen Text in die Tabelle ein.
  - `parText`
    Der Text, der eingefügt werden soll.
  - `parColSpan` (optional)
    Siehe `addElement`
  - `parColClass` (optional)
    Siehe `addElement`
- `addRemoveButton (parRemoveCB,parRemoveData,parColSpan,parColClass)`
  Fügt einen Button zum Entfernen der aktuellen Zeile in die Tabelle ein.
  - `parRemoveCB(parData)` (optional)
    Callback-Funktion, die aufgerufen wird, wenn die Zeile gelöscht wird.
- `parData`
  Daten, die beim Aufruf von `addRemoveButton` übergeben wurden.
  - `parRemoveData` (optional)
    Daten, die `parRemoveCB` beim Aufruf als Parameter übergeben werden.
  - `parColSpan` (optional)
    Siehe `addElement`
  - `parColClass` (optional)
    Siehe `addElement`
- `addImage(parImage,parColSpan,parColClass)`
  Fügt ein Bild in die Tabelle ein.
  - `parImage` URL des Bildes, das eingefügt werden soll
  - `parColSpan` (optional), siehe `addElement`
  - `parColClass` (optional), siehe `addElement`"

- `getVariables()` liefert ein Array aller Variablen der Tabelle.
  Jeder Eintrag des Arrays besteht aus einem Objekt mit den Elementen `mItemPath`, `mItemName`, `mItemHandle`.
- `addRefreshCB(parRefreshCB)` registriert eine Callback-Funktion am Tabellenobjekt, die nach Abschluss eines Refresh-Zyklus aufgerufen wird; der Callback-Funktion wird ein Array von `OPCItemValue`-Objekten übergeben.

```
OPCItemValue
{
   mItemPath
   mItemName
   mItemHandle
   mItemValue
   mItemResultId
}destructor()
```

muss aufgerufen werden, wenn die Tabelle nicht mehr benötigt wird. Gibt alle von der Tabelle belegten Ressourcen frei. Insbesondere wird die von der Tabelle verwendete Subscription bei OPC XML-DA Server abgemeldet. Die Funktion muss auch beim Verlassen der Seite aufgerufen werden (onunload).

Beispiel:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <script type="text/javascript" src="/common.js"></script>
      <script type="text/javascript" src="/opcxml.js"></script>
      <script type="text/javascript" src="/appl.js"></script>
      <script type="text/javascript">
      var gloApplDataTable = null;
      function init()
      {
          if (gloApplDataTable == null)
          {
              var tmpColumnIds     = new Array("Name_0","Value_0",
                                               "Name_1","Value_1");
              var tmpColumnClasses = new Array("Name","Value",
                                               "Name","Value")
              var tmpHeader        = new Array("Name","Value",
                                               "Name","Value");
              gloApplDataTable = new ApplDataTable(document,
                                                   "ReadTableClass",
                                                   tmpColumnClasses,
                                                   tmpColumnIds,
                                                   tmpHeader);
              gloApplDataTable.addRow("Row_0","RowClass_0");
              gloApplDataTable.addText("user1");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user1");
              gloApplDataTable.addText("user2");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user2");
              gloApplDataTable.addRow("Row_1","RowClass_1");
              gloApplDataTable.addText("user3");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user3");
              gloApplDataTable.addText("user4");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user4");
              gloApplDataTable.addRow("Row_2","RowClass_0");
              gloApplDataTable.addText("user5");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user5");
              gloApplDataTable.addText("user6");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user6");
              gloApplDataTable.addRow("Row_3","RowClass_1");
              gloApplDataTable.addText("user7");
              gloApplDataTable.addVariable("SIMOTION",
                                           "var/userdata.user7");
```

```
                gloApplDataTable.addText("user8");
                gloApplDataTable.addVariable("SIMOTION",
                                             "var/userdata.user8");
                gloApplDataTable.addRow("Time","Time");
                gloApplDataTable.addText("Time",1,"Time");
                gloApplDataTable.addVariable("SIMOTION diagnostics",
                                             "DeviceInfo.Systemtime",
                                              3,"Time");

                var tmpTableRoot =
                             document.getElementById("TableRoot");
                if (tmpTableRoot)
                {
                   tmpTableRoot.appendChild(gloApplDataTable.mTable);
                }
             }
          }
          function close()
          {
             if (gloApplDataTable != null)
             {
                 gloApplDataTable.destructor();
             }
          }
          </script>
          <style type="text/css">
            table.ReadTableClass {background-color:#FFFFE0;
                                  border:1px solid black;
                                  border-collapse:collapse;}
            th.Name {border:1px solid black;}
            th.Value {border:1px solid black;}
            td.Name {width:80px;border-right:1px solid black;}
            td.Value {background-color:#FFFFA0;width:120px;
                      text-align:right;
                      border-right:1px solid black;padding-right:15px;}
            td.Input {width:200px;}
            td.Time {border-top:1px solid black;}
            tr.RowClass_0 {background-color:#FFFFD0;border:0px;}
            tr.RowClass_1 {background-color:#FFFFB0;border:0px;}
            #Name_1Row_2 {background-color:#A0FFA0;}
          </style>
          <title>Table</title>
       </head>
       <body onload="init();" onunload="close();">
          <div id="TableRoot"></div>
       </body>
    </html>
```

Im letzten Abschnitt des Quellcodes wird mittels des <style>-Tags demonstriert, wie die Farbdarstellung von Tabellenzeilen und einzelnen Tabellenzellen mit einer CSS-Formatierung verändert werden kann.

So sorgt die Vereinbarung tr.RowClass_0 beispielsweise für die gelbe Hintergrundfarbe im Aufruf gloApplDataTable.addRow("Row_0","RowClass_0").

ApplBrowser

`ApplBrowser` ermöglicht die Abfrage des Variablenhaushalts der Steuerung.

Nach dem Anlegen eines Objektes dieses Typs wird automatisch der 1. Browsevorgang gestartet. Wenn die Browseinformationen eines Teilbaums empfangen wurden, wird zunächst die Callback-Funktion `parNewTreeFct` aufgerufen. Danach werden abhängig vom Typ der Information (Node oder Leaf) die Callback-Funktionen `parNewNodeFct` oder `parNewLeafFct` aufgerufen. Alle Callback-Funktionen erhalten als 1. Parameter ein Objekt vom Typ `ApplBrowseElement`.

Anwenderschittstelle ApplBrowseElement:

- `getElement()` liefert ein HTML-Anchor-Objekt (<a …>) für die Darstellung der Information
- `setCB(parCB)`
  Registriert eine Callback-Funktion am ApplBrowseElement-Objekt, die aufgerufen wird, wenn ein Anwender auf das Anchor-Objekt klickt. Diese Funktion erhält ebenfalls ein `ApplBrowseElement-Objekt` als Übergabeparameter.
- `destructor()` muss aufgerufen werden, wenn das Objekt nicht mehr benötigt wird. Das gilt auch für das Verlassen der Seite (`onunload`).

```
ApplBrowser(parDocument,
            parItemPath,
            parItemName,
            parNewTreeFct,
            parNewNodeFct,
            parNewLeafFct)
```

Übergabeparameter:

- `parDocument` JavaScript-Dokument, das zum Erzeugen von Elementen benutzt wird.
- `parItemPath`, `parItemName` legen den Startpunkt für das Browsen des Variablenhaushalts fest
- `parNewTreeFct(parBackElement,parItemPath,parItemName)` Callback-Funktion, die aufgerufen wird, wenn der Aufbau eines neuen Teilbaums beginnt.
  - `parBackElement` Objekt vom Typ `ApplBrowseElement`
    Der Inhalt beschreibt den Startknoten. Klickt ein Anwender auf den HTML-Anchor dieses Objektes, so wird (falls vorhanden) die Elemente des vorangehenden Teilbaums ermittelt.
  - `parItemPath,parItemName` Pfad und Name des aktuellen Teilbaums

- `parNewNodeFct(parBrowseElement)` Callback-Funktion, die für ein Node-Element aufgerufen wird
  - `parBrowseElement` Objekt vom Typ `ApplBrowseElement`
    Der Inhalt beschreibt einen Knoten. Klickt ein Anwender auf den HTML-Anchor dieses Objektes, so wird ein Browsevorgang für den mit diesem Knoten Verbundenen Teilbaum gestartet.
- `parNewLeafFct(parBrowseElement)` Callback-Funktion, die für ein Leaf-Element aufgerufen wird
  - `parBrowseElement` Objekt vom Typ `ApplBrowseElement`
    Der Inhalt beschreibt ein Blatt.

Anwenderschnittstelle:

- `destructor` gibt alle vom Browser belegten Ressourcen frei

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
    <title>Browser demo</title>
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript" src="/appl.js"></script>
    <script type="text/javascript">
      var gloBrowseTable = null;
      var gloBrowser = null;
      function addItem(parBrowseElement)
      {
        var tmpBodyElement = gloBrowseTable.firstChild;
        var tmpTableRowElement = document.createElement("tr");
        var tmpTableDataElement = document.createElement("td");
        var tmpLinkElement = parBrowseElement.getElement();
        tmpTableDataElement.appendChild(parBrowseElement.getElement());
        tmpTableRowElement.appendChild(tmpTableDataElement);
                        tmpBodyElement.appendChild(tmpTableRowElement);
      }
      function browse()
      {
        var tmpNewTreeFct =
         function(parBrowseElement,parItemPath,parItemName)
         {
            var tmpBrowseTableHook =
                      document.getElementById("BrowseTable");
            if (tmpBrowseTableHook.firstChild)
               tmpBrowseTableHook.removeChild(
                                 tmpBrowseTableHook.firstChild);
            gloBrowseTable = document.createElement("table");
            var tmpBodyElement = document.createElement("tbody");
            gloBrowseTable.appendChild(tmpBodyElement);
            tmpBrowseTableHook.appendChild(gloBrowseTable);
            if (parBrowseElement)
            {
               parBrowseElement.getElement().firstChild.nodeValue =
                  "< " +
                  parBrowseElement.getElement().firstChild.nodeValue;
               addItem(parBrowseElement);
            }
            alert("Path: " + parItemPath + "\nName: " + parItemName);
         };
         var tmpNewNodeFct = function(parBrowseElement)
         {
             parBrowseElement.getElement().firstChild.nodeValue =
                 "+ " +
                 parBrowseElement.getElement().firstChild.nodeValue;
             addItem(parBrowseElement);
```

```
            };
            var tmpNewLeafFct = function(parBrowseElement)
            {
                var tmpCB = function(parBrowseElement)
                {
                    var tmpText = "Path: " +
                                  parBrowseElement.mItemPath + "\n" +
                                  "Name: " +
                                   parBrowseElement.mItemName + "\n";
                    alert(tmpText);
                }
                parBrowseElement.setCB(tmpCB);
                addItem(parBrowseElement);
            };
            gloBrowser = new ApplBrowser(document, "", "",
                                         tmpNewTreeFct,
                                         tmpNewNodeFct,
                                         tmpNewLeafFct);
        }
        function leave()
        {
            if (gloBrowser)
                gloBrowser.destructor();
        }
    </script>
   </head>
   <body onload="browse();" onunload="leave();">
       <div id="BrowseTable"></div>
   </body>
</html>
```

ApplBrowseTree

`ApplBrowseTree` baut auf der `ApplBrowser` Klasse auf und stellt das Browseergebnis in Baumform dar.

```
ApplBrowseTree(parDocument,
               parItemPath,
               parItemName,
               parTablePrefix,
               parLeafCB,
               parNodeCB,
               parBackCB,
               parFilterCB,
               parLeafImg,
               parNodeImg,
               parBackImg)
```

Übergabeparameter:

- `parDocument`
  JavaScript document, das zum Erzeugen von Elementen benutzt wird.
- `parItemPath`, `parItemName`
  Legen den Startpunkt für das Browsen des Variablenhaushalts fest.
- `parTablePrefix`
  Präfix zum Referenzieren von Elementen in CSS
- `parLeafCB`
  Callback-Funktion, die aufgerufen wird, wenn ein Leaf angeklickt wird.
- `parNodeCB`
  Callback Funktion, die aufgerufen wird, wenn ein Node angeklickt wird. Jede der Callback Funktionen erhält als ersten Parameter ein `ApplBrowseElement`.
- `parBackCB`
  Callback-Funktion, die aufgerufen wird, wenn das erste Element des Baumes angeklickt wird.
- `parLeafImg`, `parNodeImg`, `parBackImg`
  Icons, die vor einem Back-, Node- oder Leaf-Element angezeigt werden.

Anwenderschnittstelle:

- `getElement()`
  Liefert ein Table-Element, das die Browseergebnisse beinhaltet.
- `destructor()`
  Gibt die Tabelle und alle damit verbundenen Ressourcen frei.

Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
     <title>Browse table demo</title>
     <style type="text/css">
       table.BrowseTree {table-layout:fixed;}
       td.BrowseTreeImg0 {width:20px;}
       td.BrowseTreeImg1 {width:20px;}
       td.BrowseTreeBrowse {overflow:visible;text-align:left;}
       a.refLeaf {cursor:pointer;}
       a.refNode {cursor:pointer;}
     </style>
     <script type="text/javascript" src="/common.js"></script>
     <script type="text/javascript" src="/opcxml.js"></script>
     <script type="text/javascript" src="/appl.js"></script>
     <script type="text/javascript">
     var gloBrowseTree = null;
     function browse()
     {
        var tmpLeafCB = function(parBrowseElement)
        {
            var tmpText = "Path: " + parBrowseElement.mItemPath + "\n"
                          + "Name: " + parBrowseElement.mItemName +
                          "\n";
            alert(tmpText);
        }
        var tmpFilterCB = function(parBrowseElement)
        {
            var tmpItemName = parBrowseElement.mItemName;
            var tmpPos = tmpItemName.indexOf("unit/");
            if (tmpPos != 0)
            {
                tmpPos = tmpItemName.indexOf("to/");
                if (tmpPos != 0)
                   tmpPos = tmpItemName.indexOf("var/");
            }
            return (tmpPos == 0);
         }
         gloBrowseTree = new ApplBrowseTree(document,
                                            "SIMOTION",
                                            "",
                                            "BrowseTree",
                                            tmpLeafCB,
                                            undefined,
                                            undefined,
                                            tmpFilterCB,
                                            "/ledred.gif",
                                            "/ledgreen.gif",
                                            "/ledaqua.gif");
```

```
        var tmpBrowseTreeHook =
                          document.getElementById("BrowseTree");
        tmpBrowseTreeHook.appendChild(gloBrowseTree.getElement());
      }
      function leave()
      {
          if (gloBrowseTree)
             gloBrowseTree.destructor();
      }
    </script>
  </head>
  <body onload="browse();" onunload="leave();">
    <div id="BrowseTree"></div>
  </body>
</html>
```

3.5.7 MiniWeb Server Language (MWSL)

3.5.7.1 Funktionsweise der MWSL

Bei der Webserver Language handelt es sich um eine Scriptsprache, die auf dem Webserver interpretiert wird. Sie ist der Sprache JavaScript ziemlich ähnlich, bildet aber ein kleines Subset des Sprachumfangs ab.

Durch die MWSL kann der Client mit einem einfachen Browser ohne Scripting betrieben werden, da der Webserver die Seiten dynamisch generiert.

MWSL ermöglicht Zugriff und Verarbeitung von Variablen. Unter anderem bietet es die Möglichkeit auf Prozessvariablen zuzugreifen, die auf dem System des Webservers vorhanden sind. Diese kann man dann mit MWSL und dem integrierten Template Mechanismus hervorragend aufarbeiten und entsprechend auswerten.

Der zur Erzeugung der dynamischen Seiten eingesetzte Template Mechanismus ähnelt einem sehr vereinfachten XSLT-Prozess. Siehe XSL Transformation (http://www.w3.org/standards/xml/transformation)

Der Client stellt eine Anfrage an eine URL auf dem Webserver.

Auf diesem befindet sich eine MWSL-Datei. Aus dieser wird auf dem Webserver durch den MWSL-Service eine temporäre HTML Datei generiert, welche dann zum Client gesendet und dort angezeigt wird.

3.5.7.2 Aufbau einer MWSL Datei

Grundsätzlich ist eine MWSL-Datei eine HTML-Datei, die zusätzlich MWSL Tags enthält. Zur Unterscheidung von den HTML-Dateien wird für MWSL-Dateien die Endung ".mcs" verwendet. Konvertierung von Standard HTML-Seiten in Binärdateien (Seite 122)

Beispiel:

```
<HTML>
   <HEAD>
      …
   </HEAD>
   <BODY>
      <table>
         <tr>
            [...]
            <td>
               <MWSL>
               <!—-
               //auszuführender MWSL-Code
               -->
               </MWSL>
            </td>
            [...]
            <td>
               <MWSL>
               <!—-
               //auszuführender MWSL-Code
               -->
               </MWSL>
            </td>
            [...]
         </tr>
      </table>
   </BODY>
</HTML>
```

Benötigt man die MWSL-Funktionalität, so fügt man folgende Tags hinzu:

- das <MWSL>-Tag leitet ein MWSL-Script ein
- das </MWSL>-Tag beendet das Script

Die nach dem <MWSL>-Tag vorhanden HTML-Kommentarzeichen, sind nicht unbedingt notwendig, jedoch empfehlenswert, da dadurch der MWSL-Code vom HTML Interpreter geschützt wird, und damit evtl. Fehlausgaben ausgeschlossen werden.

In den nachfolgenden Beispielen wird aus Übersichtlichkeitsgründen häufiger auf den HTML-Code verzichtet und direkt mit dem Tag "<MWSL>" begonnen.

3.5.7.3 Fehlermeldungen

Fehlermeldungen der MWSL

MWSL-Seiten mit fehlerhaften MWSL-Anweisungen enthalten einen Kommentar, der Informationen zur Fehlerursache enthält.

Im Internet Explorer kann der Quelltext einer Seite über die rechte Maustaste und den Menüpunkt **Quelltext anzeigen** in den Editor geladen werden.

Beispiel

In diesem Beispiel wird demonstriert, zu welchem Kommentar die Abfrage einer nicht vorhandenen Variable führt.

exec.mcs

```
<html>
  <head>
    <title>SIMOTION
      <MWSL>
        <!-- WriteVar("DeviceInfo.Board"); -->
      </MWSL>
    </title>
    <meta name="DC.Subject" content="SIMOTION">
    <meta name="DC.Publisher" content="Siemens AG">
    <meta name="DC.Format" content="text/html">
    <meta name="DC.Language" content="en">
    <meta name="DC.Rights" content="Copyrights Siemens AG 2003">
  </head>
  <body style="font-family: Arial">
    <p>
    <MWSL>WriteVar("var/userData.user8");</MWSL>
    </p>
    <p>
    <MWSL>WriteVar("var/userData.user9");</MWSL>
    </p>
  </body>
</html>
```

In der Datei exec.mcs wird mit dem Statement `WriteVar("var/userData.user9");` eine nicht existierende Variable abgefragt.

Quelltext der ausgegebenen Seite:

```
<html>
<head>
<title>SIMOTION
D435
</title>
<meta name="DC.Subject" content="SIMOTION">
<meta name="DC.Publisher" content="Siemens AG">
<meta name="DC.Format" content="text/html">
<meta name="DC.Language" content="en">
<meta name="DC.Rights" content="Copyrights Siemens AG 2003">
</head>
<body style="font-family: Arial">
<p>
495399
</p>
<p>

<!-- *** SCRIPT ERROR *** ---
Unknown Variable: var/userData.user9.

---- *** SCRIPT ERROR *** -->

</p>
</body>
</html>
```

Im SCRIPT ERROR-Kommentar befindet sich eine Beschreibung der Fehlerursache.

3.5.7.4 Variablentypen

MWSL unterscheidet zwischen Script-Variablen und globalen Variablen:

- Script-Variablen werden dabei innerhalb des Scripts definiert
- Globale Variablen werden von Variablenquellen zur Verfügung gestellt

Hinweis

Globale Variablen sind nicht Teil der Script Engine, sondern Informationen aus der Webserver-Umgebung. Der Zugriff auf die Variablen erfolgt ausschließlich über Zugriffsfunktionen. Die globalen Variablen sind nach ihrer Herkunft in den Variablenquellen gruppiert.

3.5.7.5 Script Variablen

Es handelt sich hierbei um Variablen, die nur auf der aktuellen Seite gültig sind.

Die Variablen gelten über MWSL-Tags hinweg, d. h. sie können in einem MWSL-Tag erstellt werden, und erst im nächsten MWSL-Tag verwendet werden.

Dabei wird nicht zwischen Variablentypen unterschieden, d. h., es gibt kein Int oder Char,…

Eine Variable wird einfach mit

`var <Variablenname> = <Wert>;` angelegt.

Der Variablentyp wird intern durch die Belegung der Variablen bestimmt.

Beispiel:

```
<MWSL>
<!--
   var string1 = "Hello";
   var string2 = "World";
   write(string1 + " " + string2);
-->
</MWSL>
```

Es werden im oben gezeigten Beispiel zwei Variablen angelegt, `string1` und `string2`.

Die beiden Strings werden (mit Leerzeichen) aneinandergehängt.

Mit dem `write` Befehl wird das Ergebnis ausgegeben (siehe `write()`).

Ausgabe: Hello World

Beispiel:

```
<MWSL>
<!--
   var num1 = 5;
   var num2 = 7;
   var Result;
   Result = num1 + num2;
-->
</MWSL>
```

Es werden im oben gezeigten Beispiel zwei Variablen angelegt, `num1` und `num2`.

Die beiden Zahlen werden addiert und das Ergebnis in der Variablen `Result` gespeichert.

`Result` enthält den Wert 12.

Die Datentyp Konvertierung erfolgt analog ECMA Script 262. Siehe Liste der Abkürzungen (Seite 271)

Schlüsselwort var

Mit dem Schlüsselwort var, wird eine Variablendeklaration eingeleitet. In ECMA-Script müssen Variablen nicht explizit deklariert werden.

Syntax:

```
varVarName = InitalValue, VarName2 = InitialValue2, …;
```

Es werden mehrere Variable deklariert und (optional) mit Initialwerten initialisiert.

Es können mehrere Deklarationen, getrennt mit Komma angegeben werden.

Nähere Informationen finden sich in der ECMA-Script Definitionen.

Sichtbarkeits- und Gültigkeitsbereiche

Die Sichtbarkeit und Gültigkeit von Variablen ist analog zu ECMA-Script. (Allerdings kennt MWSL heute keine Funktionen).

Beispiel:

```
<MWSL>
   Var MyVar = 10;
   {
      MyVar = 20;
      Write ("Inner:" + MyVar + "," );
   }
   Write ("Outer:" + MyVar + "\n" );
</MWSL>
```

Ausgabe: Inner: 20, Outer: 20

In diesem Beispiel wird in dem Anweisungsblock auf die Variable `MyVar` der äußeren Ebene zugegriffen, da auf der Ebene des Anweisungsblocks keine Variable mit dem Namen `MyVar` deklariert wurde.

Daher ändert die Anweisung `MyVar = 20` den Wert der Variablen der äußeren Ebene.

```
<MWSL>
   Var MyVar = 10;
   {
       Var MyVar = 20;
       Write ("Inner:" + MyVar + "," );
   }
   Write ("Outer:" + MyVar + "\n" );
</MWSL>
```

Ausgabe: Inner: 20, Outer: 10

3.5.7.6 Globale Variablen

Die globalen Variablen ermöglichen den Zugriff auf den Variablenhaushalt des Webserver. Es existieren drei verschiedene Arten von globalen Variablen:

- PROCESS-Variable ermöglichen den Zugriff auf die normalen Variablen des Webservers. Dies ist der Standardzugriff.
- URL-Variablen dienen dem Zugriff auf in einer URL enthaltene Variable.
- HTTP-Variablen geben den Inhalt von Variablen des HTTP-Headers zurück.

Für Testzwecke kann man diese Variablen auch durch den VarSimulator anlegen (Variablensimulator).

Der Zugriff auf eine Variable kann mit folgendem Befehl ausgeführt werden:

```
GetVar("Color", "PROCESS");
```

Die Variablenquelle `PROCESS` muss groß geschrieben werden. Ist die Variable `Color` nicht vorhanden, wird "null" zurückgeliefert.

`PROCESS` ist die Standard Variablenquelle. Daher kann `PROCESS` auch weggelassen werden.

Beispiel:

```
GetVar("Color");
```

Soll direkt ein Variablen Provider angesprochen werden, so kann anstatt der Variablenquelle `PROCESS` auch der Name des gewünschten Providers angegeben werden.

Formatstring für die Funktionen GetVar und WriteVar

Der Formatstring beginnt immer mit einem % Zeichen, gefolgt von der Angabe der Anzahl der Zeichen. Anschließend wird noch die Typangabe angegeben.

Es gibt folgende Typangaben:

- %d für Integer Werte
- %f für Float Werte
- %s für Strings

Beispiel:

"%.6s"	Gibt die ersten 6 Zeichen der angegeben Variablen aus (als String).
"%.3s"	Gibt die ersten 3 Zeichen der angegeben Variablen aus (als String).
"%.s"	Gibt die komplette Variable aus (als String).
"%3.2f"	Gibt die Variable als Float interpretiert aus. Die 3 besagt dabei, dass 3 Gesamtstellen ausgegeben werden. Die 2 gibt an, dass von den 3 Stellen 2 Nachkommastellen angezeigt werden.
"%4d"	Gibt die Variable als Integer interpretiert aus. Es werden 4 Stellen ausgegeben. Dieser Parameter kann nur übergeben werden, wenn auch die Variablenquelle "PROCESS" übergeben worden ist.

Wird der Formatstring weggelassen, so wird der komplette Inhalt der Variablen zurückgegeben.

Siehe auch

GetVar (Seite 297)

WriteVar (Seite 303)

3.5.7.7 Konfigurationskonstanten

Zugriff auf Konstanten der Konfigurationsdatei WebCfg.xml

Es besteht die Möglichkeit in der WebCfg.xml konstante Variable anzulegen. Der Zugriff auf diese Konstanten ermöglicht eine flexiblere Programmierung, so können anwenderdefinierte Seiten durch entsprechende Konfigurationsparameter gesteuert werden.

Die Definition einer Konstanten geschieht in folgendem Abschnitt der WebCfg.xml:
/SERVERPAGES/CONFIGURATION_DATA/UserConfig

```
<SERVERPAGES>
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <MyParam>MyParamValue</MyParam>
    </USERCONFIG>
  </CONFIGURATION_DATA>
</SERVERPAGES>
```

Der Zugriff in einer HTML-Seite auf die MyParam Konstante wird durch die Angabe des Pfads "constants/MyParam" ermöglicht.

```
<html>
  <head>
  </head>
  <body style="font-family: Arial">
    <table width="100%" height="100%" bgcolor="#00349A">
      <tr>
        <td style="color: #FFFFFF">
 <b><MWSL>WriteVar("constants/MyParam");</MWSL></b>
        </td>
      </tr>
    </table>
  </body>
</html>
```

Ergebnis:

```
<html>
  <head>
  </head>
  <body style="font-family: Arial">
    <table width="100%" height="100%" bgcolor="#00349A">
      <tr>
        <td style="color: #FFFFFF">
 <b>MyParamValue</b>
        </td>
      </tr>
    </table>
  </body>
</html>
```

3.5.7.8 Variablen und URL Parameter

MWSL bietet die Möglichkeit über die Funktionen `WriteVar`, `GetVar`, `SetVar` und `ExistVariable` URL-Parameterwerte zu bearbeiten.

Beispiel einer URL mit angehängten Parametern. Der Umbruch wurde zur besseren Lesbarkeit eingefügt:

```
http://localhost/MWSL/StringOperationtest.mcs?
Parameter1=Hallo&Parameter2=du!&StartValue=2&EndValue=5
```

Die URL zeigt auf die Seite StringOperationtest.mcs und übergibt die Parameter `Parameter1`, `Parameter2`, `StartValue` und `EndValue`.

Um die URL-Variable Parameter1 auszugeben, würde man z. B. folgenden Befehl angeben:

```
WriteVar("Parameter1", "URL");
```

Zu beachten ist hierbei, dass "URL" zwingend groß geschrieben werden muss.

Wird eine URL Variable angefragt, die nicht in der URL vorhanden ist, wird grundsätzlich eine Leerkette ("") zurückgegeben. Dies ist kein Script Fehler.

Parameter in URLs

In einer URL beginnt die Parameterübergabe nach dem "?"-Zeichen. Die einzelnen Parameter werden durch "&"-Zeichen getrennt. Die Wertzuweisung erfolgt nach dem "="-Zeichen.

Bestimmte Zeichen benötigen eine Kodierung, um richtig übertragen zu werden. Die folgende Tabelle gibt einen Überblick über die am häufigsten verwendeten Escape Codes.

Tabelle 3- 7 URL Escape Codes

Zeichen	Escape Code
Leerzeichen	%20
<	%3C
>	%3E
#	%23
%	%25
{	%7B
}	%7D
|	%7C
\	%5C
^	%5E
~	%7E
[	%5B
]	%5D
`	%60
;	%3B
/	%2F
?	%3F
:	%3A
@	%40
=	%3D
&	%26
$	%24

Die Kodierung setzt sich zusammen aus dem '%'-Zeichen mit folgendem ASCII-Hexadezimal-Wert des gewünschten Zeichens.

Weitere Informationen: SELFHTML (http://de.selfhtml.org)

3.5.7.9 COOKIES

Im folgenden Beispiel wird gezeigt, wie ein Cookie in einer MWSL-Datei erstellt werden kann.

Hierzu fügt man z. B. im HEAD den META-Tag:

```
http-equiv="SET-COOKIE" content="siemens_automation_language=de;"
```

Beispiel:

```
<HTML>
   <HEAD>
      <META http-equiv="SET-COOKIE"
            content="siemens_automation_language=de;">
   </HEAD>
   <BODY>
      [...]
   </BODY>
</HTML>
```

Weitere Informationen zu Cookies: http://de.selfhtml.org/

Setzen eines Cookies als HTTP Header

Es ist möglich, aus der MWSL heraus, http-Header für die http-Antwort zu setzen. Zum Beispiel kann damit das Setzen eines Cookies auch per HTTP Header und nicht als META-Tag erreicht werden.

Beispiel:

```
<MWSL>
<!--
   var strCookie;
   strCookie = "Set-cookie: siemens_automation_language=";
   strCookie = strCookie + GetVar( "Language", "URL");
   strCookie = strCookie + ", path=/\r\n";
   AddHTTPHeader( strCookie );
-->
</MWSL>
```

In diesem Beispiel wird ein Cookie zur Erkennung der eingestellten Sprache gesetzt.

3.5.7.10 Variablen und der Zugriff auf COOKIES

Der Zugriff erfolgt ähnlich wie bei den URL Parametern. Der einzige Unterschied ist, dass der Variablentyp nun `COOKIE` und nicht mehr `URL` ist.

Der Zugriff auf die im Beispiel angegebene Variable kann z. B. mit folgendem Befehl ausgeführt werden:

```
GetVar("siemens_automation_language", "COOKIE");
```

Zu beachten ist hierbei, dass `COOKIE` zwingend groß geschrieben werden muss.

Ist das `COOKIE siemens_automation_language` z. B. mit `en` gesetzt, so würde obiger Aufruf diesen Wert zurückliefern.

Weitere Informationen zu Cookies: http://de.selfhtml.org/

Ist die Variable nicht vorhanden, so gibt es keine Ausgabe.

3.5.7.11 Variablen und HTTP Header Angaben

Im Head-Tag einer HTML-Seite können verschiedenste allgemeine Informationen abgelegt werden. Mit den MWSL-Funktionen `GetVar` und `WriteVar` kann im Header gelesen und geschrieben werden.

Beispiel:

```
HTTP/1.1 GET URL\R\N
 HEADER1: WERT1\R\N
 HEADER2: WERT2\R\N
 [...]

 \R\N
```

`GetVar("HEADER1", "HTTP")` liefert den WERT1.

HTTP-HEADER können über HTML-Meta-Tags erzeugt werden.

Beispiel:

```
<HTML>
   <HEAD>
      [...]
      <META http-equiv="Accept-Language" content="de">
      [...]
   </HEAD>
   <BODY>
      [...]
   </BODY>
</HTML>
```

In diesem Beispiel wird im META-Tag die HTTP-Variable `Accept-Language` durch den Befehl "`http-equiv="Accept-Language"` definiert. Sie wird durch das Attribut `content` mit dem Wert `de` initialisiert.

Weitere Informationen zu META Angaben:

http://de.selfhtml.org/html/kopfdaten/meta.htm#allgemeines

Der Zugriff auf diese Variablen erfolgt ähnlich wie bei den URL Parametern.

Der Unterschied ist, dass die Variablenquelle `HTTP` und nicht `URL` ist.

Der Zugriff auf die im Beispiel angegebene Variable kann mit folgendem Befehl ausgeführt werden:

```
GetVar("Accept-Language", "HTTP");
```

Die Variablenquelle `HTTP` muss groß geschrieben werden.

Formatstring für die Funktionen GetVar und WriteVar

Es wird angegeben, ab welchem und wie viele Zeichen der Variablenquelle zurückgegeben werden sollen.

Dieser Parameter kann nur übergeben werden, wenn auch die Variablenquelle `HTTP` übergeben worden ist.

Wird der Formatstring weggelassen, so wird der komplette Inhalt der Variablen zurückgegeben.

Siehe auch

Globale Variablen (Seite 168)

3.5.7.12 Operatoren

Operatoren der MWSL

Alle hier vorgestellten Operatoren verhalten sich, wie in ECMA 262 definiert. Für nähere Informationen sehen Sie bitte in der ECMA 262 Spezifikation.

Die Booleschen Werte werden gegebenenfalls in die Zahlenwerte 0 (für FALSE) und 1 (für TRUE) umgesetzt.

Tabelle 3- 8 Vergleichsoperatoren

Operator	Bemerkung
<	Dieser Operator liefert TRUE zurück, wenn die linke Variable kleiner als die rechte Variable ist. Andernfalls ist der Rückgabewert FALSE.
<=	Dieser Operator liefert TRUE zurück, wenn die linke Variable kleiner oder gleich der rechten Variable ist. Andernfalls ist der Rückgabewert FALSE.
>	Dieser Operator liefert TRUE zurück, wenn die linke Variable größer als die rechte Variable ist. Andernfalls ist der Rückgabewert FALSE.
>=	Dieser Operator liefert TRUE zurück, wenn die linke Variable größer oder gleich der rechten Variable ist. Andernfalls ist der Rückgabewert FALSE.
==	Dieser Operator liefert TRUE zurück, wenn die linke Variable gleich der rechten Variable ist. Andernfalls ist der Rückgabewert FALSE.

Tabelle 3- 9 Logische Operatoren

Operator	Bemerkung
!	Logisches Nicht Dieser Operator liefert ein FALSE zurück, falls der nachfolgende Parameter TRUE ist. Ist der nachfolgende Parameter FALSE, so ist der Rückgabewert des Operators TRUE.
&&	Logisches Und Dieser Operator liefert TRUE zurück falls auf der linken und auf der rechten Seite ein TRUE-Wert steht. Andernfalls ist der Rückgabewert FALSE.
||	Logisches Oder Dieser Operator liefert ein FALSE zurück falls auf der linken und auf der rechten Seite ein FALSE-Wert steht. Andernfalls ist der Rückgabewert TRUE.

Tabelle 3- 10 Rechenoperatoren

Operator	Bemerkung
+	Plus Operator Dieser Operator addiert die linke und die rechte Variable miteinander.
=	Zuweisungsoperator Dieser Operator weist der linken Variable den Wert der rechten Variable zu.

Operator	Bemerkung
–	Minus Operator Dieser Operator zieht den Wert der rechten Variablen von der linken Variablen ab.
/	Dieser Operator teilt die linke Variable durch die rechte Variable. Der Rückgabewert ist ganzzahlig.
%	Modulo Dieser Operator gibt den Restwert einer Division zurück
*	Dieser Operator multipliziert die linke Variable mit der rechten Variablen.
++	Inkrement Dieser Operator inkrementiert (+1) die vorangestellte Variable.
--	Dekrement Dieser Operator dekrementiert (-1) die vorangestellte Variable.

3.5.7.13 For

Die MWSL bietet einen Schleifenmechanismus an, wie man ihn von JavaScript her kennt.

Für eine detaillierte Beschreibung sei hier auf die ECMA - 262 Spezifikation verwiesen.

Syntax

```
for( Startanweisung; Endbedingung; Durchlaufanweisung)
{
    Schleifenrumpf , auszuführender Code
}
```

Ablauf:

1. Die Startanweisung wird ausgeführt
2. Der Schleifenrumpf wird ausgeführt.
3. Die Durchlaufanweisung wird ausgeführt.
4. Solange die Endbedingung wahr ist, wird die Bearbeitung ab dem Schleifenrumpf (2.) wiederholt.

Beispiel:

```
for(i=1; i<5; i++)
 {
    write(i);
 }


 Ausgabe: 1234
```

3.5.7.14 If

Mit `if` wird eine Bedingung realisiert, wie sie aus Ecma - 262 bekannt ist.

Syntax

```
if(<Bedingung>)
    Anweisung1
 else
    Anweisung2;
```

Falls die Bedingung wahr ist, so wird die Anweisung 1 durchlaufen.

Ist die Bedingung nicht wahr, so wird die Anweisung 2 ausgeführt. Ist kein `else`-Teil vorhanden, wird nach der `if`-Anweisung fortgefahren.

Beispiel:

```
<MWSL>
<!--
   [...]
   if (ExistVariable("Parameter", "PROCESS"))
   {
      WriteVar("Parameter");
   }
   [...]
-->
</MWSL>
```

Falls die Prozessvariable `Parameter` existiert, so wird ihr Inhalt ausgegeben.

Falls nicht, wird die Anweisung übersprungen und die Programmausführung danach fortgesetzt. Im obigen Beispiel wäre dies der Code, der nach der schließenden geschweiften Klammer folgt.

Ist ein `else`-Zweig vorhanden, wird dieser durchlaufen, wenn die Bedingung nicht erfüllt.

Beispiel:

```
<MWSL>
<!--
   [...]
   if (ExistVariable("Parameter"))
   {
       WriteVar("Parameter");
   }
   else
   {
       write("Prozessvariable Parameter ist nicht vorhanden");
   }
   [...]
-->
</MWSL>
```

Ist die Variable Parameter nicht vorhanden, wird der `else`-Teil ausgeführt. Es wird dann eine Meldung ausgegeben, dass es keine Variable mit dem angegeben Namen gibt.

Wie die Beispiele zeigen, kann eine Anweisung durch einen Anweisungsblock ersetzt werden.

Ein Anweisungsblock ist eine Liste von Anweisungen, die in geschweifte Klammern gefasst ist.

Beispiel:

```
<MWSL>
<!--
   [...]
   if (ExistVariable("Parameter") && GetVar("Parameter")>=3)
   {
      WriteVar("Parameter");
   }
   else
   {
      write("Prozessvariable Parameter ist unzulaessig oder nicht
vorhanden");
   }
   [...]
-->
</MWSL>
```

Wenn die Prozessvariable Parameter existiert, und der Inhalt größer gleich 3 ist, so wird sie ausgegeben. Andernfalls erfolgt eine entsprechende Ausgabe.

3.5.7.15 Übersicht MWSL Funktionen

Die MWSL stellt eine Vielzahl von Funktionen zur Verfügung, die in der folgenden Tabelle als Übersicht dargestellt werden. Ausführliche Beschreibungen der Funktionen befinden sich im Anhang.

Tabelle 3- 11 MWSL Funktionen

Funktionsname	Parameter	Erläuterung
AddHTTPHeader() (Seite 294)	<Http-Header>	<Http-Header> in einer Seite einfügen.
CacheVar() (Seite 295)	<Variable>	<Variable> in den Cache-Speicher einfügen.
ExistVariable() (Seite 296)	<Variablenname>, <Variablenquelle>	Abfrage des Vorhandenseins einer Variablen.
GetVar() (Seite 297)	<Variablenname>, <Variablenquelle>, <Formatstring>	Rückgabe des Werts einer Variablen der entsprechenden Variablenquelle
InsertFile() (Seite 298)	<Textdatei>	Import einer <Textdatei>. Eine Pfadangabe ist möglich.
ProcessXMLData() (Seite 299)	<DATA>, <TEMPLATE>	Generierung dynamischer HTML-Dateien mittels spezieller XML-Dateien.
SetVar() (Seite 300)	<Variablenname>, <Wert>	Setzt Werte von Variablen.
ShareRealm() (Seite 301)	<Gruppenname>	Gibt an, ob der aktuelle User ein Mitglied der als Parameter übergebenen Gruppe ist. Der Rückgabewert kann TRUE oder FALSE sein.
write (Seite 302)	<Text>	Schreibt <Text>-Strings in die HTML-Seite. <Text> kann auch der Rückgabewert von Funktionen sein.
WriteVar (Seite 303)	<Variablenname>, <Variablenquelle>, <Formatstring>	Ausgabe eines Variablenwerts. Die Syntax ist identisch mit der GetVar() Funktion.
WriteXMLData (Seite 305)	<DATA>, <TEMPLATE>	Gibt die Daten im Gegensatz zu ProcessXMLData() direkt aus.

Tabelle 3- 12 MWSL Process-Variablen

Process-Variable	Erläuterung
NodeIndex (Seite 306)	Process-Variable für das Template parsen. Diese Variable gibt die Anzahl der bereits durchlaufenen Knoten aus.
NodeLevel (Seite 306)	Process-Variable für das Template parsen. Diese Variable gibt die Hierarchieebene des aktuellen Knotens aus.

3.5.7.16 Funktionsweise des Template Mechanismus

Ein Template ist eine Schablone, die auf Datenelemente einer Datenquelle angewendet wird. Mit diesem Mechanismus besteht die Möglichkeit, Daten und Verarbeitung getrennt voneinander zu realisieren.

Der Template Mechanismus wird durch die Befehle `ProcessXMLData()` und `WriteXMLData()` gestartet.

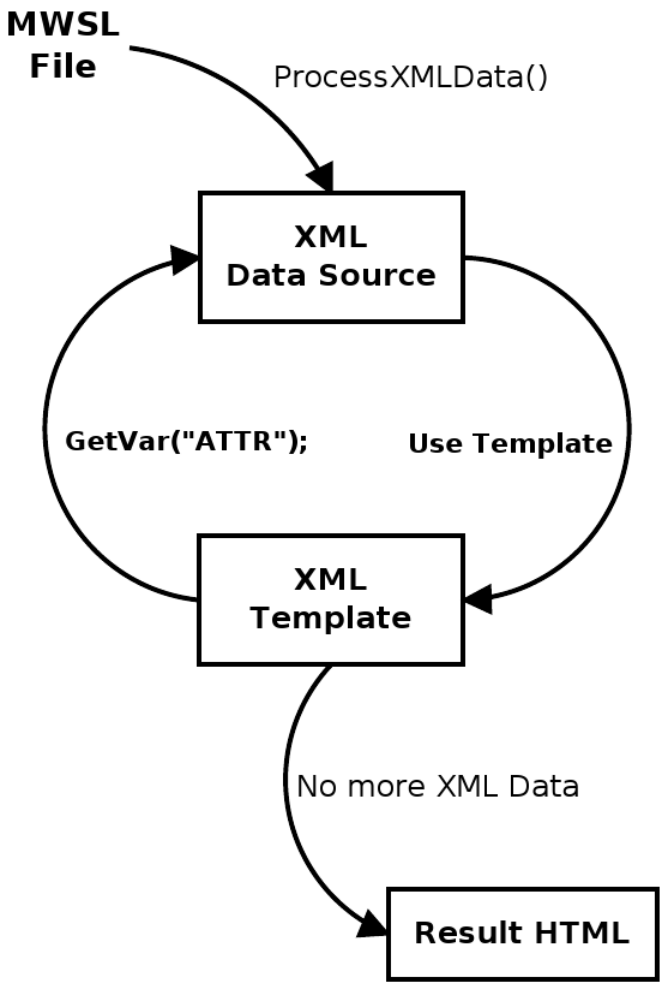


Bild 3-77 Grober Überblick des Template Mechanismus

In der Datendatei stehen strukturierte Daten, die durch den Template Mechanismus transformiert ausgegeben werden können.

Eine XML-Datei besteht aus einer Reihe von XML Knoten. Jedem XML-Knoten sind ein Name sowie eine Menge von Attributen zugeordnet. Ein Attribut besteht seinerseits aus einem Namen und einem Wert.

Die Template-Datei enthält eine Menge von Transformationsanweisungen.

Einem XML-Knoten (mit einem bestimmten Namen) kann eine Transformationsanweisung zugeordnet werden.

Beim Transformationsprozess wird das Datenfragment von oben nach unten Knoten für Knoten gelesen. Kann einem Knoten ein Template zugeordnet werden, wird dieses Template ausgeführt, die Attribute des aktuellen Knotens stehen dem Template als Variablen zur Verfügung.

3.5.7.17 Aufbau der Template-Datei

Die Template-Datei ist eine XML-Datei, deren Datenknoten die Transformationsanweisungen für die Verarbeitung einer Datendatei enthalten.

Beispiel

```
<?xml version="1.0" ?>
<TEMPLATES>
   [...]
    <TEMPLATE NAME="Variable">
       [...]
       <POSITION NAME="LINE">
       <![CDATA[
          [...]
          <MWSL>Name()</MWSL>
          [...]
       ]]>
        </POSITION>
       [...]
    </TEMPLATE>
    [...]
</TEMPLATES>
```

Die Template-Datei wertet die Datenknoten der Datendatei aus.

Im Tag `<TEMPLATES>` werden die einzelnen Template Tags definiert (im obigen Beispiel nur eins).

Mit der Zeile `<TEMPLATE NAME="Variable">` wird definiert, dass dieses Template (der nachfolgende Code) nur für Datenknoten des Typs "Variable" durchlaufen wird. Der Typ eines Datenknotens wird entweder durch den Namen des Datenknotens festgelegt, oder durch ein spezielles Attribut mit dem Namen "Template".

```
<POSITION NAME="LINE">
   [...]
</POSITION>
```

Mit diesem Tag wird festgelegt, in welchem Bereich der Webseite der Inhalt des Templates ausgegeben werden soll.

Es gibt die Positionen `HEAD`, `LINE` und `FOOT`, welche bei der Bearbeitung in der genannten Reihenfolge durchlaufen werden.

Der Inhalt der `POSITION` wird in einem `CDATA`-Block gekapselt, um ihn vor dem XML-Parser zu schützen:

```
<![CDATA[   [...]]]>
```

Mit Hilfe von MWSL kann nun auf die Attribute des aktuellen Datenknotens zugegriffen werden, und diese ausgeben oder in anderen Operationen zu verwenden.

Beispiel:

```
<MWSL> WriteVar("Name")</MWSL>
```

3.5.7.18 Aufbau einer Datenquelle

Eine Datenquelle ist ein XML-Fragment, wobei die Knoten des XML-Fragments die Datenelemente bilden.

Hinweis

Enthält das XML-Fragment keinen Root Knoten generiert MWSL selbst einen Root-Knoten, damit die Datenquelle XML-konform wird.

Das XML-Fragment kann auch ein vollständiges XML-Dokument sein.

Beispiel:

```
<?xml version="1.0" standalone="yes"?>
   [...]
   <Variable Name="ZUFUEHRUNG.STATE"
             Type="String"
             InitialValue="good"
             Behavior="Manual"
             Description="Zustand der Teilezufuehrung."
   />
   [...]
   <Variable Name="Language"
             Type="String"
             InitialValue="de_DE"
             Behavior="Manual"
             Description="Spracheinstellung der Webseite"
   />
   [...]
```

Das Beispiel definiert Datenknoten (im Beispiel `Variable`) mit den dazugehörigen Attributen.

Die unterschiedlichen Knoten können mit einer entsprechenden Template-Datei ausgewertet werden.

Eine weitere Möglichkeit besteht in der Anlage einer Datenstruktur (Hierarchie). D. h., man kann Datenknoten anlegen, die wiederum Datenknoten beinhalten.

Beispiel:

```
<?xml version="1.0" standalone="yes"?>
   [...]
   <StructVariable Name="Farbe"
                   Type="String"
                   InitialValue="gelb"
                   Behavior="Manual"
                   Description="Fiktiver Wert">
   [...]
   <Subvar Name="Rotteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="Der Rot-Anteil der Farbe"
   />
   <Subvar Name="Blauteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="Der Blau-Anteil der Farbe"
   />
   <Subvar Name="Gruenteil"
           Type="Integer"
           InitialValue="128"
           Behavior="Manual"
           Description="Der Gruen-Anteil der Farbe"
   />
   [...]
   </StructVariable>
   [...]
```

In diesem Beispiel gibt es den Datenknoten vom Typ `StructVariable`. Dieser Datenknoten beinhaltet mehrere Datenknoten vom Typ `Subvar`.

Für die unterschiedlichen Typen von Datenknoten können unterschiedliche Templates verwendet werden.

3.5.7.19 Template Transformation

Der Befehl `ProcessXMLData()` generiert aus einer XML-Datendatei und einer XML-Template-Datei dynamisch eine HTML-Datei (oder auch nur ein Textfragment).

Hierzu geht der Parser von oben nach unten Schritt für Schritt durch die Datendatei.

Er liest einen Datenknoten ein. Dann wird in der Template-Datei nach einem passenden Template für diesen Datenknoten gesucht. Wird ein Template gefunden, wird das Template auf den Datenknoten angewendet. Das Template ist ein MWSL-Fragment. Der einzige Unterschied ist, dass die Attribute des XML Datenknotens im Template, wie normale Prozessvariablen zur Verfügung stehen, wenn keine Variablenquelle angegeben wird. Bei Namensgleichheit überdecken die XML-Attribute entsprechende Prozessvariablen. Wird explizit die Variablenquelle PROCESS angegeben, dann werden immer die Prozessvariablen verwendet.

Beispiel:

```
Prozessvariable
Color Value "Green"
```

Datendatei:

```
<?xml version="1.0" standalone="yes"?>
[...]
<Variable Name="ZUFUEHRUNG.STATE"
          Farbe="Red"
/>
[...]
<Variable Name="Language"
/>
[...]
```

Template-Datei:

```
<?xml version="1.0" ?>
<TEMPLATES>
   <TEMPLATE NAME="Variable">
      <POSITION NAME="LINE">
      <![CDATA[
         <MWSL>write (GetVar("Name")+":");</MWSL>
         <MWSL>write (GetVar("Color")+"\r\n");</MWSL>
      ]]>
      </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

Ausgabe:

```
ZUFUEHRUNG.STATE: Red
```

```
Language: Green
```

Ablauf eines Template-Prozesses:

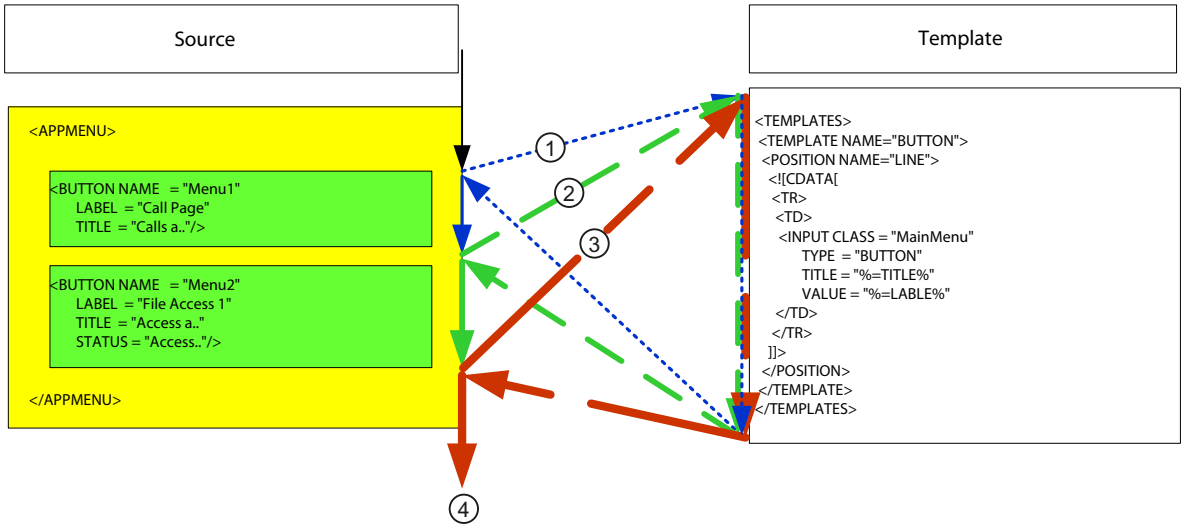


Bild 3-78 Ablauf des Template-Prozesses

Das Bild zeigt den Ablauf des Template Parse-Prozesses.

Für jedes Datenelement (gelbe und grüne Blöcke) wird jeweils einmal das Template durchlaufen. Die Attribute des aktuellen Datenelements sind hierbei als Variablen erreichbar.

Zeitliche Abfolge:

1. Im Source wird das Tag `<APPMENU>` gefunden und das Template durchlaufen. Im Template wird keine Verarbeitungsanweisung für dieses Tag gefunden, deshalb wird kein Datenelement in die Ausgabe geschrieben.
2. Das erste Tag `<BUTTON>` wird gefunden und vom Template verarbeitet.
3. Das zweite Tag `<BUTTON>` wird gefunden und vom Template verarbeitet.
4. Es werden keine weiteren Tags gefunden, die Verarbeitung ist beendet.

Ein weiteres Beispiel:

aufrufende MWSL Datei

```
<HTML>
   <HEAD>
      <TITLE>
         MWSL Template Test Page
      </TITLE>
   </HEAD>
   <BODY >
   <MWSL>
   <!--
      write(ProcessXMLData(
            "<EXTERNAL SRC=\"/MWSL/variables.xml \"/>",
            "<TEMPLATES><EXTERNAL SRC=\"/MWSL/variablesTemplate.xml\"/>
            </TEMPLATES>")
      );
   -->
   </MWSL>
   </BODY>
</HTML>
```

variables.xml

```
<?xml version="1.0" standalone="yes"?>
 <Provider Name ="MyVarProvider">
   <Variable Name="ZUFUEHRUNG.STATE"
             Type="String"
             InitialValue="good"
             Behavior="Manual"
             Description="Zustand der Teilezufuehrung."
   />
   <Variable Name="Language"
             Type="String"
             InitialValue="de_DE"
             Behavior="Manual"
             Description="Spracheinstellung der Webseite"
   />
 </Provider>
<!-- End of File -->
```

variablesTemplate.xml

```
<?xml version="1.0" ?>
<TEMPLATES>
   <TEMPLATE NAME="Provider">
      <POSITION NAME="HEAD">
      <![CDATA[
         <TABLE BORDER="1">
            <TR>
               <TH>
                  Variablenname
               </TH>
            </TR>
      ]]>
      </POSITION>
      <POSITION NAME="FOOT">
      <![CDATA[
            <TR>
               <TD>
                  <MWSL><!--GetVar("Name");--></MWSL>
               </TD>
            </TR>
         </TABLE>
      ]]>
      </POSITION>
   </TEMPLATE>
   <TEMPLATE NAME="Variable">
      <POSITION NAME="LINE">
      <![CDATA[
         <TR>
            <TD>
               <MWSL>
               <!--
                  GetVar("Name");
               -->
               </MWSL>
            </TD>
         </TR>
      ]]>
      </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

Durch Aufruf des Befehls `ProcessXMLData` beginnt der Parser mit der Datendatei variables.xml. Das `<Provider>`-Tag wird zuerst gefunden.

In der Template-Datei variablesTemplate.xml wird gesucht, ob ein Template für diesen Typ definiert ist.

Der Bereich `Position="HEAD"` wird ausgeführt.

Der Parser liest das nächsten Tag aus der Datendatei und generiert anhand des entsprechenden Templates in der Template-Datei die weiteren Zeilen der auszugebenden HTML-Datei.

Gleiches passiert mit dem nächsten Tag.

Mit dem End-Tag `</Provider>` wird der Fußteil des Template Providers ausgeführt.

Im nachfolgenden Ausdruck ist der generierte Teil fett geschrieben.

Generierte Datei:

```
<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>
  <BODY >
 <TABLE BORDER="1">
 <TR>
 <TH>Variablenname</TH>
 </TR>
 <TR>
 <TD>ZUFUEHRUNG.STATE</TD>
 </TR>
 <TR>
 <TD>Language</TD>
 </TR>
 <TR>
 <TD>Hallo</TD>
 </TR>
 </TABLE>
  </BODY>
</HTML>
```

3.5.7.20 MWSL in XML Attributen

Im Rahmen des Template-Parsens ist es sinnvoll, auch in XML Attributen der Datendatei MWSL-Statements zu schreiben, die dann zum Parse Zeitpunkt ausgewertet werden.

Beispiel:

```
<?xml version="1.0" standalone="yes"?>
<Motor Name="M1"
       Nummer="1"
       Type="Dreh"
       Nennleistung = "7"
       Drehzahl="3"
       Alter="2"
       Farbe="RED"
       Prozess="&MWSL;WriteVar(&quot;CPULoad&quot;,&quot;PROCESS&quot;,
                               &quot;%e&quot;); &END_MWSL;"
/>
```

Im Beispiel wird der folgende MWSL-Befehl benutzt:

```
"&MWSL;WriteVar(&quot;CPULoad&quot;,&quot;PROCESS&quot;,&quot;%e&quot;);
&END_MWSL;"
```

Der Befehl würde in einer Template-Datei oder in einer MWSL-Datei so aussehen:

```
<MWSL>WriteVar("CPULoad", "PROCESS", "%e");</MWSL>
```

Dieser Befehl gibt den Wert der Variablen `CPULoad` der Variablenquelle `PROCESS` aus.

Dabei ist zu beachten, dass `<MWSL>` durch `&MWSL;` und `</MWSL>` durch `&END_MWSL;` ersetzt werden müssen. Des Weiteren sind doppelte Anführungszeichen `"` durch `&quot;` zu ersetzen.

Der Rest entspricht der MWSL-Syntax.

3.5.7.21 Beispiele

Beispiele für den Einsatz der MWSL

Die hier gezeigten Beispiele verdeutlichen die Möglichkeiten der MWSL.

Variablenwerte mit der Funktion SetVar setzen

```
<MWSL>
   SetVar(GetVar("VARNAME"), GetVar(GetVar("VARNAME"), "URL"));
</MWSL>
```

In diesem Beispiel wird die Variable, deren Name in der Prozessvariablen `VARNAME` gespeichert ist, mit dem Wert der URL-Variablen `VARNAME` initialisiert.

Zur Verdeutlichung:

`GetVar("VARNAME")` liefert den Inhalt der Prozessvariablen `VARNAME`. Dieser Wert wird wiederum als Variablenname gesehen.

Nimmt man an, dass die Prozessvariable `VARNAME` "Willibald" als Inhalt hat, so sieht der gesamte Aufruf schon wesentlich einfacher aus:

Gesamter Aufruf: `SetVar("Willibald", GetVar("Willibald", "URL"));`

Angenommen die URL-Variable "Willibald" hätte den Inhalt "ist ein toller Kerl", dann wäre dies äquivalent zum folgenden Ausdruck:

```
SetVar("Willibald", "ist ein toller Kerl");
```

TestTemplate.mcs

Mit diesem Beispiel wird noch einmal kurz der Template-Mechanismus erläutert.

Zur Veranschaulichung sind einige Absätze in den jeweiligen Dateien markiert.

Das Template erzeugt eine Tabelle.

Die aufgerufene Datei

TestTemplate.mcs

```
<HTML>
   <HEAD>
      <TITLE>
         MWSL Template Test Page
      </TITLE>
   </HEAD>
   <BODY>
   <MWSL>
   <!--
      write(ProcessXMLData("<EXTERNAL SRC=\"/MWSL/variables.xml\"/>",
         "<TEMPLATES><EXTERNAL SRC=\"/MWSL/variablesTemplate.xml\"/>
         </TEMPLATES>")
      );
   -->
   </MWSL>
   </BODY>
</HTML>
```

VariablesTemplate.xml

```
<?xml version="1.0" ?>
<TEMPLATES>
   <TEMPLATE NAME="Provider">
      <POSITION NAME="HEAD">
      <![CDATA[
         <TABLE BORDER="1">
            <TR>
               <TH>
                  Varname
               </TH>
               <TH>
                  Type
               </TH>
               <TH>
                  Description
               </TH>
               <TH>
                  Value
               </TH>
               <TH>
                  NI/NL
               </TH>
           </TR>
      ]]>
      </POSITION>
      <POSITION NAME="FOOT">
      <![CDATA[
         <TR>
            <TD COLSPAN="5">
            </TD>
         </TR>
      </TABLE>
      ]]>
      </POSITION>
   </TEMPLATE>

   <TEMPLATE NAME="Variable">
      <POSITION NAME="LINE">
      <![CDATA[
         <TR>
            <TD align=center>
               <A HREF="<MWSL>Link()</MWSL>"><MWSL>Name()</MWSL></A>
            </TD>
            <TD>
               <MWSL>Type()</MWSL>
            </TD>
            <TD>
               <MWSL>Description()</MWSL>
            </TD>
            <TD align=right>
```

```
                <MWSL>InitialValue()</MWSL>
              </TD>
              <TD>
                <MWSL> NodeIndex() </MWSL> / <MWSL> NodeLevel() </MWSL>
              </TD>
            </TR>
        ]]>
        </POSITION>
    </TEMPLATE>
</TEMPLATES>
```

Die Template-Datei besteht in diesem Fall aus mehreren Templates. Das Template `Provider` stellt den Kopf und Fußteil für die Daten (Tabellenkopf und Fuß).

Das Template `Variable` ist für die Zeilen der Datenausgabe bestimmt (pro Variableneintrag eine Tabellenzeile).

Die Daten werden nach den entsprechenden Attributen eingefügt.

Die Datendatei `variables.xml`

```
<?xml version="1.0" standalone="yes"?>
<Provider Name ="MyVarProvider">
 <Variable Name="ZUFUEHRUNG.STATE"
 Type="String"
 InitialValue="good"
 Behavior="Manual"
 Description="Zustand der Teilezufuehrung."
 />
 <Variable Name="Language"
 Type="String"
 InitialValue="de_DE"
 Behavior="Manual"
 Description="Spracheinstellung der Webseite"
 />
</Provider>
```

generierte HTML Datei:

```
<HTML>
   <HEAD>
      <TITLE>
         MWSL Template Test Page
      </TITLE>
   </HEAD>
   <BODY>
      <TABLE BORDER="1">
         <TR>
            <TH>
               Varname
            </TH>
            <TH>
               Type
            </TH>
            <TH>
                Description
            </TH>
            <TH>
               Value
            </TH>
            <TH>
               NI/NL
            </TH>
         </TR>
        <TR>
           <TD align=center>
              <A HREF="">ZUFUEHRUNG.STATE</A>
           </TD>
           <TD>
              String
           </TD>
           <TD>
              Zustand der Teilezufuehrung.
           </TD>
           <TD align=right>
              good
           </TD>
           <TD>
              2 / 2
           </TD>
        </TR>
        <TR>
           <TD align=center>
              <A HREF="">Language</A>
           </TD>
           <TD>
              String
           </TD>
           <TD>
```

```
                Spracheinstellung der Webseite
              </TD>
              <TD align=right>
                 de_DE
              </TD>
              <TD>
                 3 / 2
              </TD>
           </TR>
        </TABLE>
     </BODY>
</HTML>
```

MainNavigation.mcs

```
<html>
   <head>
      <title>
         MiniWeb Main Navigation
      </title>
   </head>
   [...]
   <body>
      <table>
         <tr>
         <MWSL>
             write(ProcessXMLData(
               "<EXTERNAL SRC=\"/XML/MainNavigation.xml\"/>",
               "<TEMPLATES><EXTERNAL
                      SRC=\"/Templates/MainNavigation.xml\"/>
               </TEMPLATES>") );
         </MWSL>
         </tr>
      </table>
   </body>
</html>
```

Diese Datei beinhaltet den eigentlichen Rumpf der zu generierenden HTML Seite.

Der dynamische Teil wird mit dem Befehl `ProcessXMLData()` generiert, und in `<table>` eingefügt.

/XML/MainNavigation.xml

MainNavigation.xml beinhaltet den Datenteil für die Generierung.

```
<?xml version="1.0" standalone="yes"?>
<MAINNAVIGATION>
   <APPLICATION NAME = "Entrance"
                CLIENTAREA = "/Portal/Entrance.mcs"
                TITLE = "Back to Entrance Page." />
   <APPLICATION NAME = "MWSL Test"
                CLIENTAREA = "/MWSL/Start.mcs"
                TITLE = "Test environment for MWSL."/>
   <APPLICATION NAME = "File Browser"
                       SECUREGROUP = "Administrator"
                       CLIENTAREA = "/www"
                       TITLE = "Browse the Filesystem" />
   [...]
   <APPLICATION NAME = "CSSA"
                SECUREGROUP = "User"
                CLIENTAREA = "/CSSA/Main.mcs"
                TITLE = "PKI Interface."/>
                <APPLICATION NAME = "VarSimulator"
                CLIENTAREA = "/Simulator/Simulator_index.mcs"
                TITLE = "Simulate several variables."/>
</MAINNAVIGATION>
```

/Templates/MainNavigation.xml

```
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
   <TEMPLATE NAME="APPLICATION">
   <POSITION NAME="LINE">
   <![CDATA[
      <td>
         <input class = "MainMenu"
                type = "BUTTON"
                title = "<MWSL> WriteVar("TITLE")</MWSL>"
                value = "<MWSL> WriteVar("NAME")</MWSL>"
                OnClick =
                 "NavigateApp('<MWSL>WriteVar("CLIENTAREA")</MWSL>')"
         />
      </td>
   ]]>
   </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

Diese Datei wird für jeden Datenknoten durchlaufen und die entsprechenden Variablen werden eingefügt.

generierte HTML Datei

```
<html>
   <head>
      <title>
         MiniWeb Main Navigation
      </title>
   </head>
   <body>
      <table>
          <tr>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Back to Entrance Page."
                       value = "Entrance"
                       OnClick = "NavigateApp('/Portal/Entrance.mcs')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Test environment for MWSL."
                       value = "MWSL Test"
                       OnClick = "NavigateApp('/MWSL/Start.mcs')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Browse the Filesystem"
                       value = "File Browser"
                       OnClick = "NavigateApp('/www')"
                />
             </td>
             [...]
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "PKI Interface."
                       value = "CSSA"
                       OnClick = "NavigateApp('/CSSA/Main.mcs')"
                />
             </td>
             <td>
                <input class = "MainMenu"
                       type = "BUTTON"
                       title = "Simulate several variables."
                       value = "VarSimulator"
                       OnClick = "NavigateApp
                                   ('/Simulator/Simulator_index.mcs')"
                />
```

```
            </td>
         </tr>
      </table>
   </body>
</html>
```

AppNavigation.mcs

Aufruf:

```
<MWSL>
   WriteXMLData("<EXTERNAL SRC=\"" + GetVar("XML", "URL") + "\"/>",
                "<TEMPLATES><EXTERNAL SRC=\"" +
                GetVar("TEMPLATE", "URL") + "\"/></TEMPLATES>");
</MWSL>
```

Beim Aufruf werden die Daten und die Template-Datei aus der URL übernommen.

Als übergebene Template-Datei wird in diesem Beispiel AppNavigation.xml angenommen, als übergebene Datendatei MWSLTestMenu.xml.

MWSLTestMenu.xml

```
<?xml version="1.0" ?>
<APPMENU>
   <MENU>
      <BUTTON NAME = "Menu1"
              LABEL = "Variablentest"
              TITLE = "Tooltip"
              STATUS = "Statusline"
              CLIENTAREA = "/MWSL/Variablentest.mcs?Parameter=4711"
      />
      [...]
      <BUTTON NAME = "Menu9"
              LABEL = "Versuch"
              TITLE = "Tooltip"
              STATUS = "Statusline"
              CLIENTAREA = "/MWSL/Versuch.mcs" />
   </MENU>
</APPMENU>
```

In der nachfolgenden Template-Datei werden einige Attribute nach ihrer Existenz abgefragt und die entsprechenden Zeilen ausgeführt.

In diesem Fall wird nur die Bedingung für `CLIENTAREA` durchlaufen, da kein anderes abgefragtes Attribut vorhanden ist.

AppNavigation.xml

```
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
   <TEMPLATE NAME="BUTTON">
      <POSITION NAME="LINE">
      <![CDATA[
         <TR>
            <TD>
               <INPUT CLASS = "MainMenu"
                      TYPE = "BUTTON"
                      TITLE = "<MWSL>TITLE()</MWSL>"
                      VALUE = "<MWSL>LABEL()</MWSL>"
                  <MWSL>
                  if ( ExistVariable( "CLIENTAREA" ))
                  {
                     write("OnClick=
                       \"top.ClientArea.window.navigate('"+
                           GetVar("CLIENTAREA") + "')\"");
                   }
                   if ( ExistVariable ( "TOP" ))
                   {
                       write("OnClick = \"top.window.navigate('" +
                        GetVar("TOP") + "')\"");
                    }
                    if ( ExistVariable ( "WIN" ))
                    {
                        write("OnClick = \"window.open('" +
                                 GetVar("WIN") + "')\"");
                    }
                    if ( ExistVariable ( "ACTION" ))
                    {
                       write("OnClick = \"top.ClientArea.window." +
                         GetVar("ACTION") + "\"");
                    }
                    </MWSL>
                 />
              </TD>
          </TR>
      ]]>
      </POSITION>
   </TEMPLATE>
</TEMPLATES>
```

In der Template-Datei befinden sich `if`-Bedingungen, in denen Attribute aus der Datendatei abgefragt werden.

Somit ergeben sich für jeden Datenknoten zur Laufzeit unterschiedliche Anweisungen.

In der generierten HTML Datei wird dann nur die dem jeweiligen Datenknoten entsprechende Zeile stehen.

generierte Ausgabe

```
<TR>
   <TD>
      <INPUT CLASS = "MainMenu"
             TYPE = "BUTTON"
             TITLE = "Tooltip"
             VALUE = "Variablentest"
             OnClick = "top.ClientArea.window.navigate(
                         '/MWSL/Variablentest.mcs?Parameter=4711')"
      />
   </TD>
</TR>
[...]
<TR>
   <TD>
      <INPUT CLASS = "MainMenu"
             TYPE = "BUTTON"
             TITLE = "Tooltip"
             VALUE = "Versuch"
             OnClick =
               "top.ClientArea.window.navigate('/MWSL/Versuch.mcs')"
      />
   </TD>
</TR>
```

3.5.8 Anwenderdefinierte Verzeichnisseite

Im IT DIAG werden verschiedene Aufgaben in der Default Applikation zusammengefasst, die über das Tag <DEFAPP> in der WebCfgFrame.xml konfiguriert werden können.

Die Default Applikation hat die folgenden Aufgaben:

- Senden von Dateien
  - HTML-Seiten
  - Images
  - Etc.
- Empfangen einer neuen WebCfg.xml Konfigurationsdatei
  - Parsen dieser Datei
  - Restart des Webservers
- Generieren des Verzeichnis Browsers
  - Erstellen der Verzeichnisinhalte
  - Löschen von Dateien
  - Erstellen bzw. Löschen von Verzeichnissen
  - Laden von Dateien

Für einige dieser Aufgaben muss die Default Applikation selbst HTML-Seiten generieren und zum Client zurücksenden, z. B. für den Verzeichnis-Browser, oder die Bestätigung, wenn eine neue Datei empfangen wurde.

Diese Seiten sollen frei gestaltbar sein, damit sie sich in das gewünschte Aussehen des Zielsystems einbinden lassen.

Hierzu werden mehrere HTML-Fragmente definiert, mit deren Hilfe die Default Applikation die Antwort Seiten zusammenstellt. Diese HTML-Fragmente liegen in dem Konfigurationsbereich.

Die HTML-Fragmente müssen in `<![CDADA[ .. ]]>` Blöcke gekapselt werden, da sie unter Umständen keine gültige XML-Syntax aufweisen.

Übersicht:

```
<SERVERPAGES>
   [...]
   <CONFIGURATION_DATA>
      <DEFAPP>
         <DIRHEAD>
            [...]
         </DIRHEAD>
         <DIRTAIL>
            [...]
         </DIRTAIL>
         <DIRLINE>
            [...]
         </DIRLINE>
         <LOADSUCCEED>
            [...]
         </LOADSUCCEED>
         <SENDCOMPLETE>
            [...]
         </SENDCOMPLETE>
      </DEFAPP>
      [...]
   </CONFIGURATION_DATA>
 [...]
</SERVERPAGES>
```

Es werden fünf solcher HTML-Fragmente bzw. HTML-Seiten definiert:

- **<**DIRHEAD**>** Verzeichnis-Kopf
- **<**DIRLINE**>** Verzeichniseintrag (eine Zeile)
- **<**DIRTAIL**>** Verzeichnis-Fuß
- **<**LOADSUCCEED**>** Bestätigungsseite, dass eine neue Konfigurations-XML-Datei empfangen wurde.
- **<**SENDCOMPLETE**>** Bestätigungsseite, dass eine neue Datei empfangen wurde.

Aufbau einer Verzeichnisseite

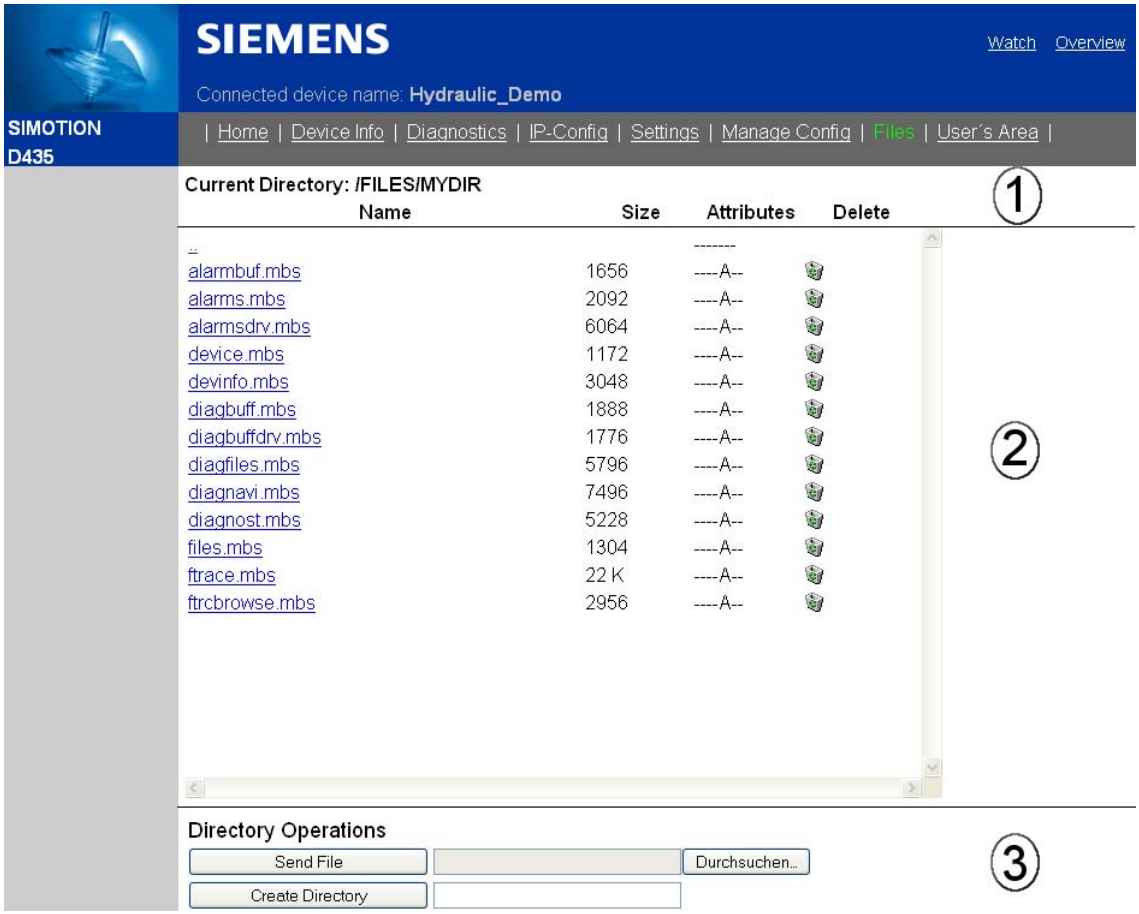


Bild 3-79 Bereiche einer Webseite

Der Verzeichnis Browser wird aus 3 Abschnitten zusammengesetzt:

- einem Kopfbereich ① `<DIRHEAD>`
- einem Datenbereich (der die Verzeichnisdienste enthält) ② `<DIRLINE>`
- und einem Fußbereich ③ `<DIRTAIL>`

Zusammengesetzt ergeben die drei Fragmente eine HTML-Seite.

Das bedeutet, dass der Kopfbereich ein `<HTML>`-Tag enthält aber kein schließendes `</HTML>`-Tag, dieser ist im Fußbereich zu finden. Ansonsten dürfen der Kopf- und der Fußbereich beliebige HTML-Konstrukte enthalten.

Für jeden Verzeichniseintrag wird jeweils ein `<DIRLINE>` Fragment eingefügt.

Der Verzeichnisbrowser stellt einige Variablen bereit, die in den HTML-Fragmenten verwendet werden können:

%=DIRPATH%	Liefert den absoluten Pfad des gerade angezeigten Verzeichnisses. Hierbei wird der Pfad aus Browser Sicht gezeigt, also wie er in der URL steht, nicht der physikalische Pfad im lokalen Dateisystem. Letzteres ist aus Sicherheits- und technischen Gründen nicht möglich.
%=ICONPATH%	Kann nur im <DirLine> Bereich verwendet werden. Erzeugt einen <IMG Src="…"> für die aktuelle Datei. Als Icon wird ein passendes Icon aus der Mime-Type-Tabelle ermittelt (Anhand der Dateiendung der Datei). Siehe Dateitypen <MIME_TYPES>. In obigem Beispielbild Sieht man die Explorer-Kugel als Icon bei den HTML-Seiten, bzw. ein Ordnersymbol bei den Verzeichnissen. Diese sind das Resultat des %=ICONPATH% Eintrages.
%=DESTINATIONPATH%	Kann nur im <DirLine> Bereich verwendet werden. Erzeugt eine URL über die die aktuelle Datei angefordert werden kann. Diese Variable kann z. B. dazu verwendet werden, einen Link auf diese Datei anzulegen.
%=FILENAME%	Kann nur im <DirLine> Bereich verwendet werden. Liefert den Namen der aktuellen Datei.
%=FILESIZE%	Kann nur im <DirLine> Bereich verwendet werden. Liefert die Größe der aktuellen Datei.
%=FILEATTRIBUTES%	Kann nur im <DirLine> Bereich verwendet werden. Liefert eine Zeichenkette mit den Attributen der aktuellen Datei: R: Die Datei ist schreibgeschützt. H: Die Datei ist versteckt. S: Die Datei ist eine Systemdatei. A: Die Datei muss archiviert werden, wurde also seit der letzten Sicherung geändert. F: Die Datei ist im ROM (Fix). Ist ein Attribut bei der Datei nicht gesetzt, wird an der entsprechenden Stelle ein "-" eingetragen.

%=TRASHBINPATH%	Erzeugt ein <IMG>-Tag mit dem Icon für einen Mülleimer (der Pfad wird aus der Mime-Type-Tabelle, Eintrag: "trashbin" erzeugt.). Wird das Icon angeklickt, wird eine Script Funktion "Delete()" aufgerufen. Diese erhält als Parameter die URL zu der aktuellen Datei, erweitert um den Parameter "?DELETE". In dieser Script Funktion (Die z. B. im <DIRHEAD> Teil definiert werden kann) Kann z. B. nach Sicherheitsabfrage diese URL angefordert werden, woraufhin die Datei von der Default Applikation gelöscht werden würde. (Ausreichende Rechte des Benutzers vorausgesetzt.) Wird eine Datei zum Server gesendet, so muss die Default Applikation diesen Upload mit einer HTML-Antwort bestätigen. Diese HTML-Antwort wird unter dem Tag <SENDCOMPLETE> abgelegt. Der Inhalt dieser HTML-Datei kann eine entsprechende Bestätigung enthalten, oder auch den Auftrag das geänderte Verzeichnis neu anzuzeigen.
%=FORWARDURL%	Kann nur im <SENDCOMPLETE> Bereich verwendet werden. Liefert eine URL, mit der das geänderte Verzeichnis (erweitert um die gerade geladene Datei) angezeigt werden kann. Wird z. B. im Body der HTML-Seite nur ein: <SCRIPT> location.replace("%=FORWARDURL%"); </SCRIPT> eingetragen, wo wird automatisch die Verzeichnisseite geladen.

In den Fragmenten können auch zusätzlich MWSL Statements verwendet werden.I

Bestätigungsseiten

Im Tag <LOADSUCCEED> wird die Antwort für einen erfolgreichen WebCfg.xml Upload hinterlegt.

Mit dem Tag <SENDCOMPLETE> kann die weitere Verarbeitung nach einem Upload oder Delete gesteuert werden.

3.5.9 Server Side Includes (SSI)

3.5.9.1 Einbindung von Prozesswerten

In den anwenderdefinierten HTML-Seiten können Sie Prozesswerte mittels Server Side Includes (SSI) einbinden.

Hinweis

Ab der SIMOTION Steuerung V4.1 müssen HTML-Seiten mit SSI als Binärdatei vorliegen, um Prozesswerte darzustellen. Eine Standard HTML-Seite kann mit dem mitgelieferten Konvertierungstool (Seite 122) in eine Binärdatei umgewandelt werden.

HTML-Seiten mit ausschließlich statischem Inhalt müssen nicht konvertiert werden.

Einbindung von Prozesswerten

Die Einbindung in der HTML-Seite erfolgt durch die Zeichenfolge `<%=IDENTIFIER %>`. `IDENTIFIER` ist ein Platzhalter, den Sie durch Variablen der Variablen Provider ersetzen müssen. Zum Beispiel `<%=DeviceInfo.Board%>`, diese Variable liefert die Bezeichnung der Steuerung. Auf einer D435 ist es z. B. der Wert "D435".

Details zu den Variablen und zur Syntax können Sie im Kapitel Variablen Provider (Seite 239) nachschlagen.

Im folgenden Quelltext sehen Sie ein Beispiel für die Einbindung der Variablen `userData.user1`. Zunächst wird der Wert der Variablen ausgegeben (Systemvariable `userData.user1`: `<%=var/userData.user1 %>`). Im Eingabefeld wird der Wert der Variablen als Vorbelegung verwendet und wird, sofern eine Eingabe erfolgt, überschrieben.

```
<html>
    <head>
        <title>Demo Seite</title>
    </head>
    <body text="#000000" bgcolor="#FFFFFF" link="#FF0000" alink="#FF0000"
    vlink="#FF0000">
        Demoseite<br>
        Systemvariable userData.user1 : <%= var/userData.user1 %> <br>
        <form method="post" action="/VarApp">
                SIMOTION C: userData.user1:
                <input type="TEXT" name="var/userData.user1" value="<%=
                var/userData.user1 %>" />
                <input type="submit" value="Wert schreiben" />
        </form>
    </body>
</html>
```

3.6 OPC XML-DA Server

3.6.1 Überblick

Der SIMOTION IT OPC XML-DA Server ermöglicht via Ethernet auf Daten und Betriebszustände des SIMOTION Geräts zuzugreifen.

Was ist OPC XML-DA?

OPC ist die Abkürzung für open connectivity und bezeichnet eine Standardschnittstelle für die Kommunikation in der Automatisierungstechnik.

Mit OPC XML-DA ist es möglich, über Ethernet basierende Standardtelegramme mit einer Steuerung zu kommunizieren. Zur Befehlsübermittlung dient das Kommunikationsprotokoll SOAP (simple object access protocol).

Die Schnittstelle selbst wird in einer Konfigurationsdatei mittels einer Beschreibungssprache, die auf XML-Vokabular basiert, definiert. Sie beschreibt das Format der Anforderungs- und Antwort-Nachrichtenströme, mit denen dann Funktionsaufrufe abgesetzt werden (siehe OPC XML-DA R1.0 Specification http://www.opcfoundation.org/Downloads.aspx).

Die Nutzung dieser Schnittstelle erfordert grundsätzlich eine Applikation als Client.

Folgendes Bild stellt einen Client dar, der von der OPC Foundation zur Verfügung gestellt wird.

Der Client ermöglicht das Browsen über die System-, Interface-, IO- und geräteglobalen-Variablen.

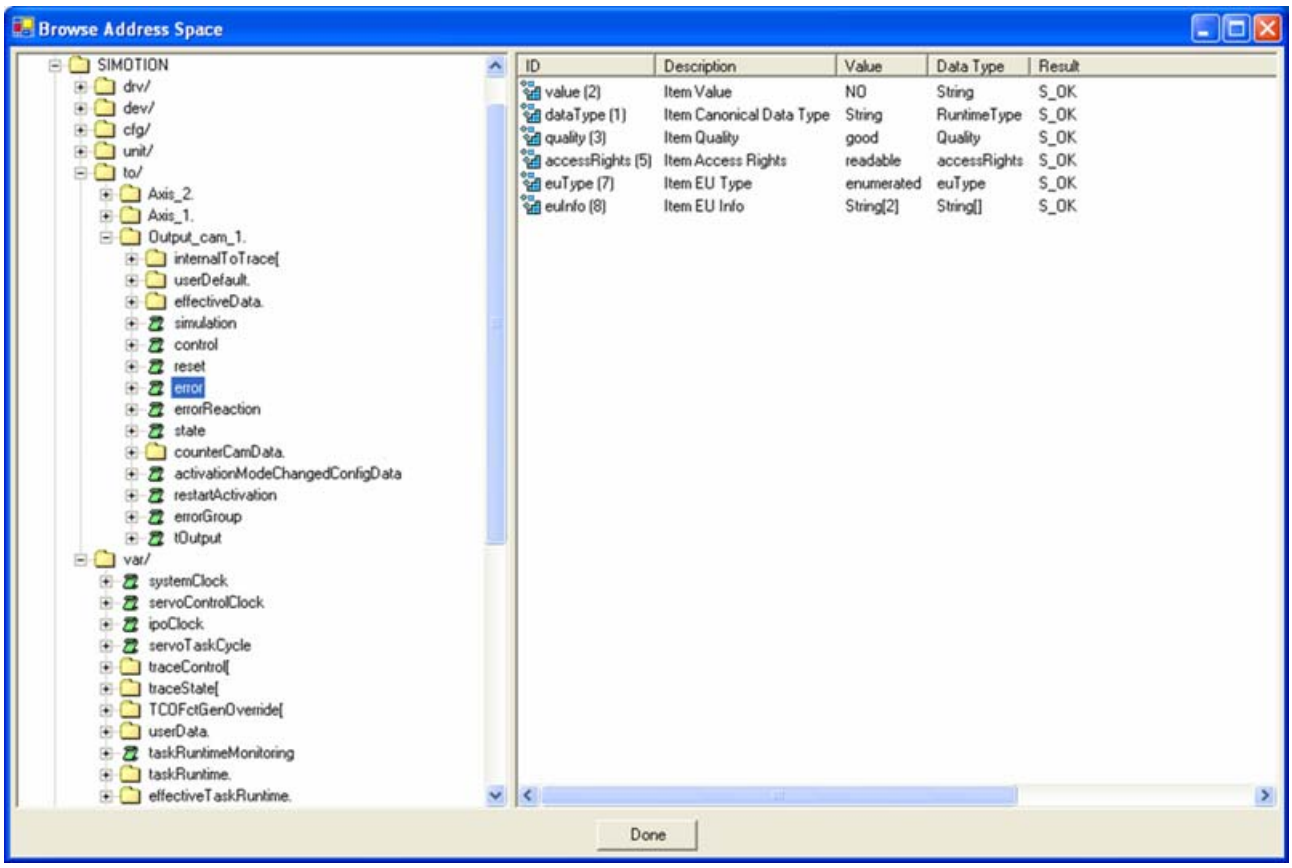


Bild 3-80 Client

Ziel und Nutzen

Das Ziel und der Nutzen des SIMOTION IT OPC XML-DA Servers ist folgender:

- Gemäß OPC XML-DA V1.0 Standard, über Ethernet-Schnittstelle ansprechbar.
- Der Server kann von beliebigen Client-Applikationen, die OPC XML-DA V1.0 konform sind, angesprochen werden, unabhängig von deren Betriebssystem (z. B. Linux).

Welche Vorkenntnisse sind notwendig?

Zum Verständnis des in diesem Kapitel beschriebenen SIMOTION IT OPC XML-DA Servers wird vorausgesetzt, dass der Anwender mit der Begriffswelt von OPC XML-DA (siehe OPC XML-DA R1.0 Specification) vertraut ist.

3.6.2 Gegenüberstellung OPC XML-DA/SIMATIC NET OPC-DA

Gegenüberstellung

Neben dem SIMOTION IT OPC XML-DA Server existiert noch das Produkt SIMATIC NET OPC-Server für SIMOTION. Dieses Paket ermöglicht über SIMATIC NET OPC-DA ebenfalls auf Daten und Betriebszustände des SIMOTION Geräts zuzugreifen.

Folgende Tabelle stellt diese beiden Pakete gegenüber und beschreibt die prinzipielle Vorgehensweise:

Tabelle 3- 13 Prinzipielle Vorgehensweise beim Zugriff auf Daten

SIMOTION IT OPC XML-DA	SIMATIC NET OPC-DA
Keine Projektierung (OPC-Export) mit SIMOTION SCOUT nötig. Programmvariable über Schalter aktivierbar.	OPC-Export mit SIMOTION SCOUT erforderlich, zu wiederholen bei jeder Projektänderung.
Symbole werden erst im SIMOTION Gerät aufgelöst, Kommunikation per Textformat (XML).	Symbole werden beim OPC-Export aufgelöst und im OPC-Server auf Windowssystem binär hinterlegt, Kommunikation binär-> hoher Datendurchsatz.
Derzeit nur SIMOTION mit OPC XML-DA. Zugriff auf S7-Geräte derzeit noch nicht möglich.	Gleichzeitiger Zugriff auf SIMOTION und S7-Geräte möglich.
Client läuft auf beliebigen Betriebssystemen.	Basiert auf Windows COM/-DCOM-Technik, Client und Server laufen nur auf Windowsbetriebssystemen.
Kommunikation über Standardprotokolle (TCP/IP, XML, SOAP), keine herstellerspezifischen (SIEMENS) Tools, Treiber auf Clientsystem notwendig.	Verwendung S7-Protokoll zur Kommunikation, auf Client-Seite entsprechende herstellerspezifische Treiber notwendig.
Kommunikation nur über Ethernet möglich.	Kommunikation über PROFIBUS/MPI und Ethernet möglich.
Direkte Adressierung über Firewalls möglich.	DCOM - Kommunikation wird in der Regel an Firewalls nicht freigegeben.

3.6.3 Prinzipielle Darstellung zur Designzeit

Beispielhafte Anordnung

Folgendes Bild veranschaulicht beispielhaft die Anordnung der jeweiligen Software bei der Erstellung einer Client-Applikation auf einem PC. Der PC und das SIMOTION Gerät sind vernetzt über Ethernet.

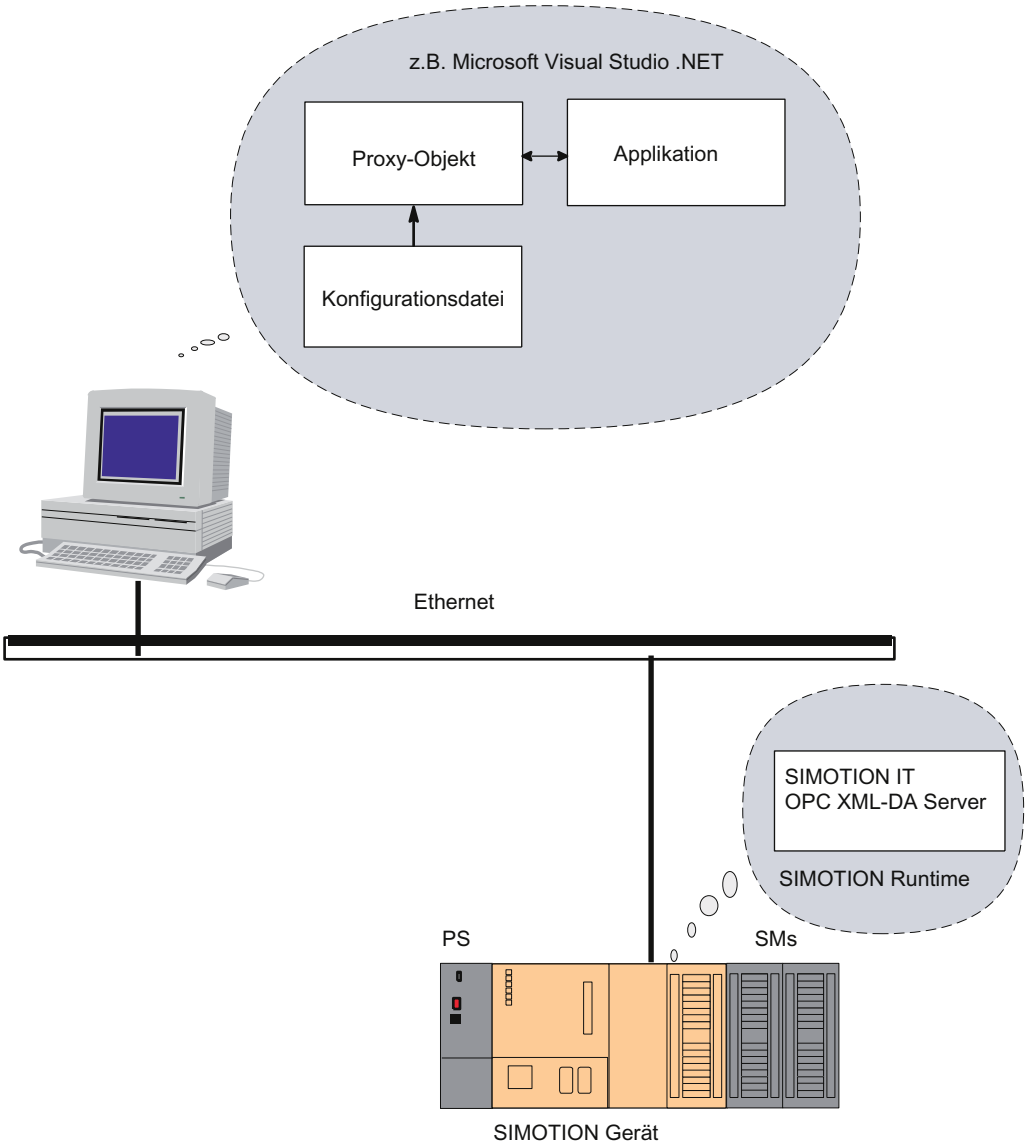


Bild 3-81 Übersicht Designzeit (Beispiel)

3.6.4 Prinzipielle Darstellung zur Laufzeit

Beispielhafte Anordnung

Folgendes Bild veranschaulicht beispielhaft die Anordnung der Client-Applikation auf einem PC und des OPC XML-DA Servers auf dem SIMOTION Gerät während der Laufzeit. Beide sind vernetzt über Ethernet.

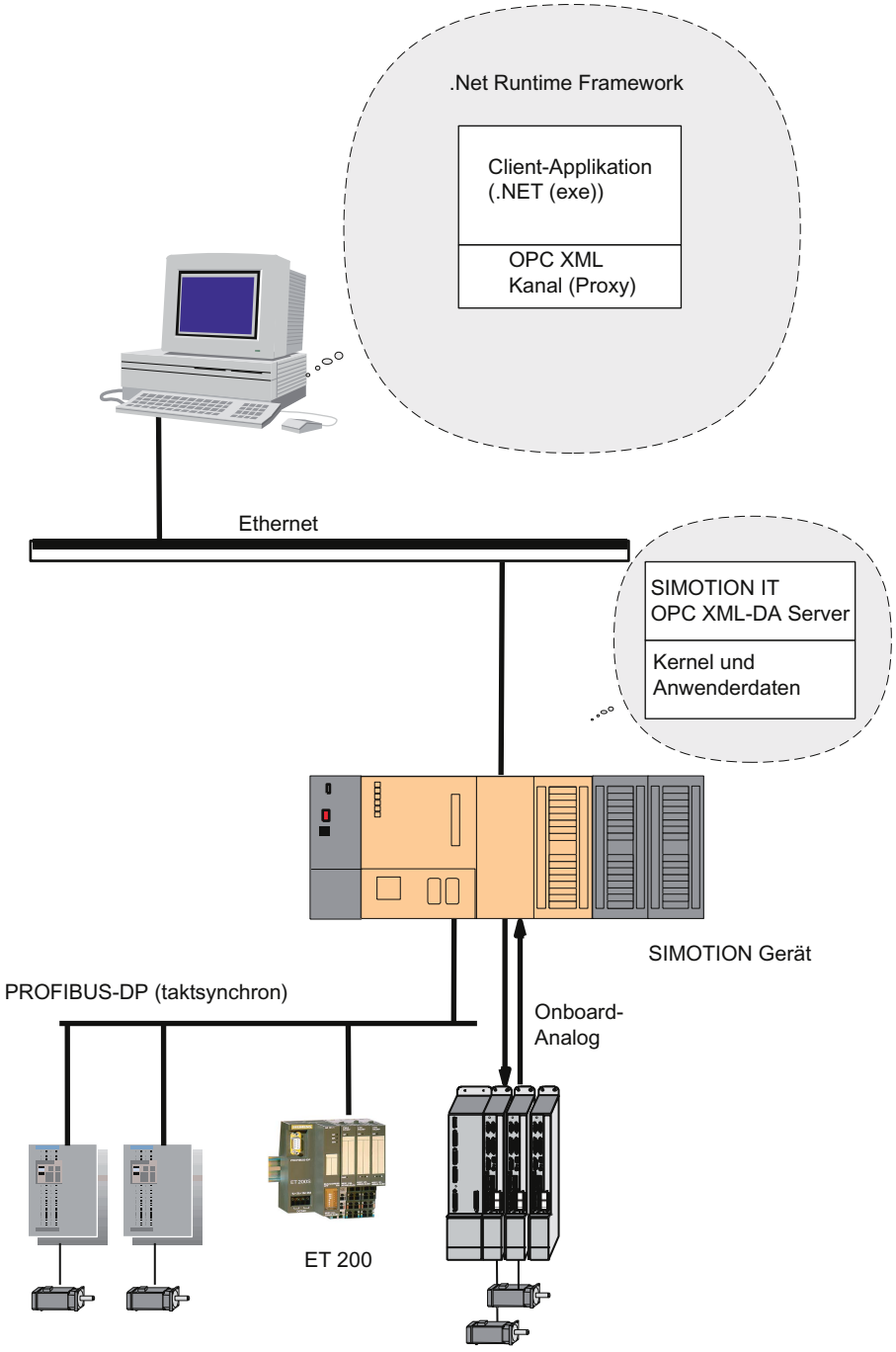


Bild 3-82 Übersicht Laufzeit (Beispiel)

3.6.5 Installation

3.6.5.1 Hard- und Softwarevoraussetzungen zur Designzeit

Hardwarevoraussetzungen zur Designzeit

Hinweis

Die Auswahl für die Programmierumgebung ist frei wählbar. Die nachfolgend aufgeführten Voraussetzungen sind beispielhaft für Microsoft Visual Studio .NET, jedoch nicht bindend.

Tabelle 3- 14 Hardwarevoraussetzungen zur Designzeit

Ausbau	Mindestanforderung
Prozessor	Intel Pentium III oder kompatibel, 800 MHz
Hauptspeicher	128 MByte RAM

Softwarevoraussetzungen zur Designzeit

Hinweis

Die Auswahl für die Programmierumgebung ist frei wählbar. Die nachfolgend aufgeführten Voraussetzungen sind beispielhaft für Microsoft Visual Studio .NET, jedoch nicht bindend.

- Microsoft Visual Studio .NET:

  http://msdn.microsoft.com/vstudio/

  http://www.microsoft.com/net/
- Konfigurationsdatei (WSDL), gemäß OPC XML- R1.0 Specification.

3.6.5.2 SIMOTION Geräte-Schnittstelle zur Laufzeit konfigurieren

Schnittstelle konfigurieren

Um zur Laufzeit einen Verbindungsaufbau von einem PC zu einem SIMOTION Gerät herzustellen, müssen folgende Schritte zur Konfiguration der Ethernet-Schnittstelle ausgeführt werden:

Tabelle 3- 15 Schnittstelle konfigurieren

Schritt	Vorgehen
1	Die Funktionalität muss im SIMOTION Projekt bei der Hardwarekonfiguration der Steuerung über die Eigenschaften "Ethernet erweitert" in der Funktion "OPC XML/Diagnoseseiten" aktiviert sein.
2	Die Adresse des Servers muss bekannt sein. Diese kann über die Schnittstelleneinstellungen in HW-Konfig konfiguriert werden.

3.6.6 Unit-Variablen verfügbar machen

Damit Unit-Variablen im SIMOTION IT OPC XML-DA Server verfügbar sind, müssen Sie diese als VAR_GLOBAL deklarieren.

Unit-Variablen im Interface deklarieren

In der Deklarationstabelle legen Sie für die jeweilige Variable den Typ fest. Nur als VAR_GLOBAL deklarierte Variablen stehen im OPC XML-DA zur Verfügung.

Folgende Abbildung zeigt exemplarisch die Deklaration von Unit-Variablen in einem MCC-Programm.

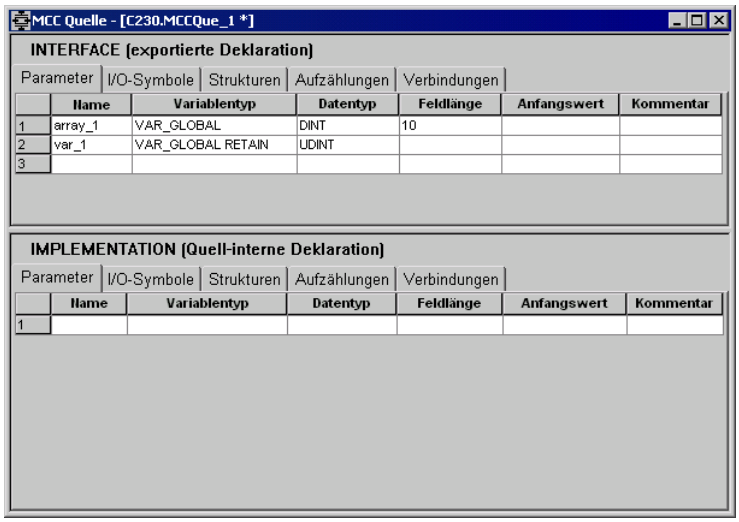


Bild 3-83 Globale Variable deklarieren

OPC-XML ermöglichen

Für die Aktivierung der Variablen für OPC XML-DA gehen Sie folgendermaßen vor:

1. Öffnen Sie den Dialog **Programmeigenschaften**.
2. Öffnen Sie das Register **Compiler**.
3. Aktivieren Sie **OPC-XML ermöglichen**, falls dies noch nicht der Fall ist (Standardeinstellung).

Folgende Abbildung zeigt das Aktivieren einer Unit-Variablen aus einem MCC-Programm.

Bild 3-84 Variablen für OPC XML-DA verfügbar machen

Hinweis

Die Aktivierung von OPC-XML gilt auch für Variablen in Kop/Fup und ST-Programmen. In einem ST-Programm müssen Sie die Variablen, die für OPC XML-DA verfügbar sein sollen, in einem globalen Variablenblock definieren. Dieser muss im Interfaceabschnitt stehen.

3.6.7 Beispiel für eine Client-Applikation

Beispiel

Das folgende Beispiel erläutert die wichtigsten Programmierschritte für die Methode "Read" mit dem Tool Microsoft Visual Studio .NET2003.

Die folgende Anwendung führt den OPC-Client aus:

Die Beispielanwendung zeigt in einer Dialogbox eine Schaltfläche **Read**. Mit dem Betätigen dieser Schaltfläche verbindet sich der Client mit dem SIMOTION IT OPC XML-DA Server und liest eine Variable. Das Ergebnis wird im Ausgabefeld des Dialoges angezeigt.

Das folgende Bild zeigt die Dialogbox der Beispielanwendung:

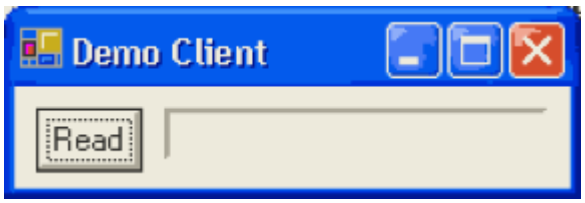


Bild 3-85 Demo Client

Programmierschritte

Folgende Programmierschritte sind notwendig:

1. Sie erstellen mit Microsoft Visual Studio .NET ein neues Projekt und importieren die WSDL-Datei als Schnittstellenbeschreibung (Menü "Add Web Reference").
2. Sie erstellen eine Dialogbox mit einem Textfeld und einem Button **Read**.
3. Den für die Referenz vergebenen Namen z. B. "OPCXMLServer", machen Sie im Programm bekannt (`using DemoClient.OPCXMLServer`).
4. Machen Sie die Server URL wie folgt im Programm bekannt:

   http://<IP-Adresse>/soap/opcxml

   Fügen Sie unter <IP-Adresse> die IP-Adresse ihrer SIMOTION Steuerung ein.
5. Sie instanziieren gemäß dem Beispielcode das Server-Proxy-Objekt und versorgen den Aufruf mit den erforderlichen Parametern.
6. Als Rückgabe erhalten Sie die angeforderten Daten.

Programmausschnitt

```
using DemoClient.OPCXMLServer;
private void ReadButton_Click(object sender, System.EventArgs e)
{
     Service SIMOTION_C_Server = new Service();


     RequestOptions ReadOptions = new RequestOptions();
     ReadOptions.ClientRequestHandle = "";
     ReadOptions.LocaleID = "DE-AT";
     ReadOptions.RequestDeadlineSpecified = false;
     ReadOptions.ReturnDiagnosticInfo = false;
     ReadOptions.ReturnErrorText = false;
     ReadOptions.ReturnItemName = false;
     ReadOptions.ReturnItemPath = false;
     ReadOptions.ReturnItemTime = false;


     RequestOptions ReadOptions = new RequestOptions();
     ReadOptions.ClientRequestHandle = "";
     ReadOptions.LocaleID = "DE-AT";
     ReadOptions.RequestDeadlineSpecified = false;
     ReadOptions.ReturnDiagnosticInfo = false;
     ReadOptions.ReturnErrorText = false;
     ReadOptions.ReturnItemName = false;
     ReadOptions.ReturnItemPath = false;
     ReadOptions.ReturnItemTime = false;


     ReadRequestItemList ReadItemList = new ReadRequestItemList();
     ReadRequestItem[] ReadItemArray = new ReadRequestItem[1];
     ReadRequestItem ReadItem = new ReadRequestItem();


     ReadItem.ItemPath = "SIMOTION";
     ReadItem.ItemName = "var/userData.user5";
     ReadItemArray[0] = ReadItem;
     ReadItemList.Items = ReadItemArray;


     ReplyItemList ReadReplyList;
     OPCError[] ReadErrorList;


     SIMOTION_C_Server.Url = "http://simotion/soap/opcxml";
     System.Net.ICredentials myCredentials = new
     System.Net.NetworkCredential("simotion","simotion");
     SIMOTION_C_Server.Credentials = myCredentials;
     SIMOTION_C_Server.PreAuthenticate = true;
     System.Net.ServicePointManager.Expect100Continue = false;


     SIMOTION_C_Server.Read(ReadOptions,ReadItemList,out ReadReplyList,out
     ReadErrorList);


     if ((ReadReplyList.items[0] != null) &&
          (ReadReplyList.Items[0].Value != null) &&
          (ReadReplayList.Items[0].Value.GetType().Name != "XmlNode[]"))
          Output.Text = ReadReplyList.Items[0].Value.ToString();
     else Output.Text = "<Error>";

}
```

Hinweis

Wenn mit Microsoft Visual Studio .NET die Client-Applikation erstellt wird, dann muss mit dem Menü "Add New Item" ein "Application configuration file" der Solution hinzugefügt werden. Diese Textdatei ist notwendig, um die Debuginformation im SOAP-Telegramm abzuschalten.

Der "Application configuration file" muss folgenden Inhalt haben:

Application configuration file

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <system.diagnostics>
                <switches>
                        <add name="Remote.Disable" value="1"/>
                </switches>
        </system.diagnostics>
</configuration>
```

3.6.8 Schnittstelle SIMOTION IT OPC XML–DA Server

3.6.8.1 Übersicht

Einleitung

Dieses Kapitel beschreibt die Methoden, die Sie über die Schnittstelle zum OPC XML-DA V1.0 Server ausführen können. Der Server selbst ist im SIMOTION Gerät integriert.

Vor der Version 4.2 musste der Server über eine Lizenzierung freigeschaltet werden.

Es ist ein kurzer Überblick. Ausführlich beschrieben sind diese Methoden in dem Dokument "OPC XML-DA Specification R1.0" der OPC Foundation.

Die aktuelle ausführliche Schnittstellenbeschreibung finden Sie auf der Homepage der OPC Foundation: http://www.opcfoundation.org

3.6.8.2 Synchron aufrufbare Methoden

Der SIMOTION IT OPC XML-DA Server bietet folgende synchron aufrufbare Methoden unter dem Typ "OpcXmlDaService" an:

Beschreibung der Methoden

Browse

Die Methode "Browse" ermöglicht das Navigieren über die vorhandenen Variablen.

GetProperties

Die Methode "GetProperties" kann zu einer bestimmten Variablen Einstellungen abfragen (z. B. Zugriffsrechte, Zeitstempel, Datentyp).

GetStatus

Die Methode "GetStatus" liefert Informationen zum Status des Servers, der Programmversion und der unterstützten Interfaceversion.

Read

Die Methode "Read" liest Variablenlisten aus.

Subscribe

Die Methode "Subscribe" übergibt eine Liste mit Variablennamen, und erhält ein Handle dieser Subscription. Dieses Handle kann in der Methode SubscriptionPolledRefresh verwendet werden, um die Werte der vorher definierten Variablen wieder zu pollen. Siehe Prinzipielles zu Subscriptions (Seite 226)

Die Eigenschaft Buffering, die dem Attribut "EnableBuffering" entspricht, wird nicht unterstützt. Das betrifft die Methoden "SubscribeRequestItem" und "SubscribeRequestItemList".

SubscriptionPolledRefresh

Mit der Methode "SubscriptionPolledRefresh" bekommen Sie die Werte der Variablen, welche zuvor mit der Methode Subscribe beschrieben wurden. Das Handle, das die Subscription liefert, wird als Parameter verwendet.

Der Parameter "Holdtime" legt den frühesten Antwortzeitpunkt fest. Damit wird die Häufigkeit der Datenübertragung begrenzt.

Mit dem Parameter "ReturnAllItems" wird die Verwendung des Parameters "WaitTime" bestimmt.

- True

  "WaitTime" wird ignoriert, alle geforderten Werte werden sofort zurückgegeben.

- False

  Für die Dauer der im Parameter "WaitTime" angegebenen Zeit prüft der Server, ob sich einer der angeforderten Werte seit dem letzten Aufruf geändert hat.

  Verstreicht die angegebene Zeit, ohne dass sich Werte verändern, wird eine leere Antwort zurückgegeben.

  Ändern sich während der angegebenen Zeit Werte, so werden die geänderten Werte sofort zurückgeschickt und die Abfrage ist beendet.

SubscriptionCancel

Die Methode "SubscriptionCancel" beendet die Subscription und gibt das Subscriptionhandle zurück.

Beim Aufruf muss angegeben werden, welche Subscription beendet werden soll.

Wenn eine asynchrone Form des Aufrufes verwendet wird, dann wird der Client zu einem späteren Zeitpunkt über ein Client Handle informiert, welche Subscription beendet wurde.

Hinweis

Nachdem die Subscription beendet ist, ist für den Client das Subscriptionhandle ungültig.

Write

Die Methode "Write" schreibt Variablenlisten.

3.6.8.3 Variablenzugriff

Variablenzugriff über Methoden

Über die synchron und asynchron aufrufbaren Methoden kann auf Variablen zugegriffen werden.

Hinweis

Informationen für den Zugriff auf Unit-Variablen in einem MCC-Programm erhalten Sie im Programmierhandbuch SIMOTION MCC.

Die Informationen für einen Zugriff in einem ST-Programm erhalten Sie im Programmierhandbuch SIMOTION ST.

Siehe auch

Unit-Variablen verfügbar machen (Seite 214)

3.7 Trace Interface via SOAP (TVS)

Einleitung

Der auf SOAP basierende Service bietet die Möglichkeit eines Trace-Services an.

Der WebTrace ist identisch mit dem SIMOTION SCOUT Trace. Der einzige Unterschied ist das unterschiedliche Format der ausgegebenen Daten.

Trace-Service

Das Funktionspaket "Trace Interface via SOAP" ermöglicht es, Variablenwerte in einen Puffer zu schreiben. Die Werte werden in Dateien gepackt und können per HTTP-Request asynchron abgeholt werden.

Die Nutzung dieser Schnittstelle erfordert grundsätzlich eine Applikation als Client. Der Client ermöglicht die Aufzeichnung eines zeitlichen Verlaufes von Variablen.

Zum Erstellen der Applikation existiert eine WSDL-Datei.

3.7.1 Trace Interface via SOAP

Einleitung

Bei der Arbeit mit einem Trace nimmt dieser verschiedene Zustände ein. Die folgende Grafik zeigt die möglichen Zustände und Übergänge. Die genannten Methoden werden im Kapitel "Schnittstelle Trace" beschrieben.

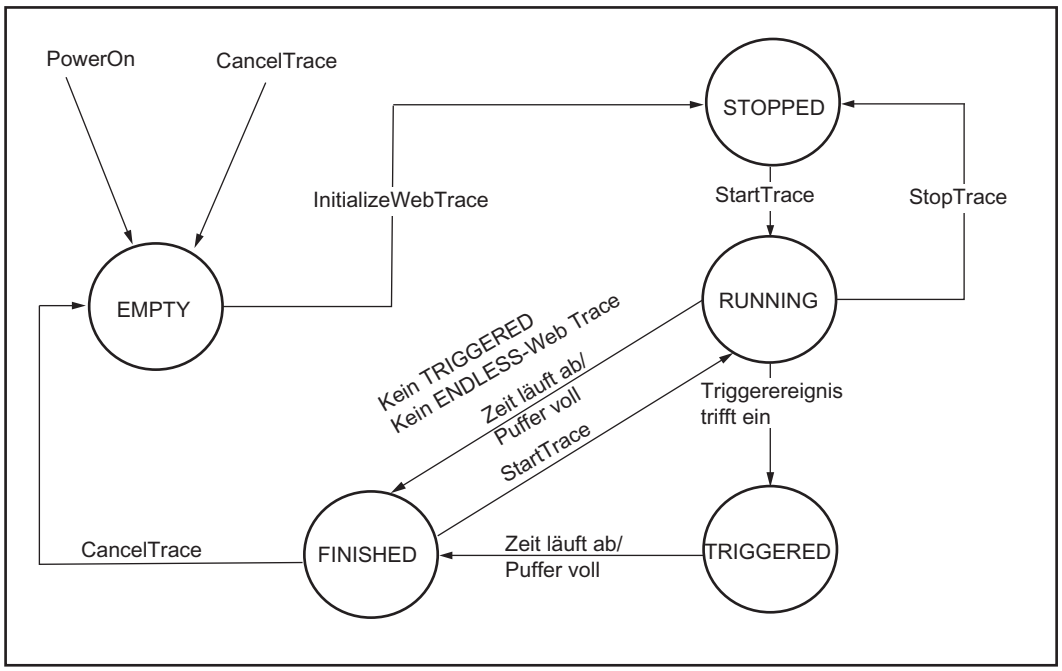


Bild 3-86 WebTrace

Zustände

Nachdem ein Trace mit "InitializeWebTrace" angelegt wurde, steht dieser auf STOPPED. Mit StartTrace läuft er an und schreibt die gewünschten Daten in den Puffer. Dementsprechend kann er auch mit "StopTrace" wieder gestoppt werden. Nach dem Start geht der Trace in den Zustand RUNNING. Ist die im Aufruf spezifizierte Zeit abgelaufen, dann nimmt der Trace den Zustand FINISHED ein. Zu jeder Zeit kann ein Trace mit "CancelTrace" gelöscht werden, um ihn z. B. neu anzulegen.

3.7.2 Vorgehensweise/Begriffe

HTTP Methoden - Datenaustausch

Die Tracedaten werden mit der Methode ReadData als zusammengefasste Daten auf der RAM-Disk abgelegt. Diese sind über gewöhnlich HTTP-Requests abzuholen.

Hinweis

Die Daten werden durch das alleinige Abholen nicht gelöscht! Um einen Überlauf der RAM-Disk zu verhindern, muss nach einem HTTP-GET Aufruf ein HTTP-DELETE Aufruf auf diese URL erfolgen. (Hintergrund: Es ist der UseCase angedacht, dass ein Client u. U. dieselben TraceDaten mehrmals anfordern muss, z. B. um verschiedene bereits gelaufene Traces miteinander zu vergleichen.) Erst nach einem CancelTrace werden diese temporären Daten komplett gelöscht, egal ob bereits abgeholt oder nicht.

TRIGGERED

Der Trace bietet die Möglichkeit der Triggerung. Hierfür sind je nach Triggermethode unterschiedliche Konstanten bzw. Variablensymbole anzugeben. Der Trace startet mit:

- Positive Flanke (RE),
  wenn die Variable den Wert einer Konstanten übersteigt.
- Negative Flanke (FE),
  wenn die Variable den Wert einer Konstante unterschreitet.
- Innerhalb eines Toleranzbandes (WIB),
  wenn die Variable zwischen zwei Konstanten liegt.
- Außerhalb eines Toleranzbandes (OOB),
  wenn die Variable außerhalb eines Toleranzbandes liegt.
- Bitmaske hat Wert (BHV),
  wenn die Variable nach der Maskierung mit einer Konstanten einen vorgegebenen Wert hat.

Wird der Trace im Modus TRIGGERED eingerichtet, ist eine Triggerbedingung wie weiter unten beschrieben, anzugeben. Dieser Trigger fungiert als SingleShot. Durch den Parameter MatchCountTriggerPoint kann er aber auf wiederholtes Auftreten gesetzt werden (z. B. fünf: Erst beim fünften Eintreten der Triggerbedingung triggern).

Hier findet die Datenaufzeichnung erst nach der Triggerung statt. Die Daten werden für die Dauer aufgezeichnet, die beim Einrichten angegeben wurde.

IMMEDIATE / ENDLESS

Das Gegenstück zum TRIGGERED Trace ist der IMMEDIATE Trace, der sofort nach Eintreffen des StartTrace Aufrufs die Aufzeichnung beginnt. Auch hier werden die Daten für die Dauer aufgezeichnet, die beim Einrichten angegeben wurde.

Eine Ringpufferaufzeichnung verwendet der ENDLESS Trace. Triggerbedingungen werden nicht ausgewertet. ENDLESS Trace startet, sobald das StartTrace-Ereignis kommt. Er wird allerdings nur beendet, wenn explizit StopTrace aufgerufen wird. Die Größe des Ringpuffers ist ebenfalls über die Dauer beim Initialisierungs-Aufruf anzugeben. Somit ist ein sinnvoller Wert zu finden, der ressourcenschonend ist, aber ausreicht, um per HTTP die Daten rechtzeitig abzuholen.

Die Größe des Ringpuffers (B) ergibt sich aus der Anzahl der Variablen (N), ihrer Größe (S), der übergebenen Zeitdauer (t) und dem Takt (T), in dem sie aufgezeichnet werden.

$$B = t/T * \sum_{i=0}^{n-1} S^i$$

Innerhalb der übergebenen Zeitdauer muss der Puffer durch Aufruf der Funktion "readData" mindestens einmal entsorgt werden, um ein Überschreiben der jeweils ältesten Trace-Daten zu verhindern.

Sollten die Parameter einen größeren Speicherbereich erfordern, so wird die Aufzeichnungsdauer so reduziert, dass nicht mehr als 256 KByte belegt werden.

3.7.3 Fehlerbehandlung

Alle implementierten Methoden des TVS (Trace Via SOAP) liefern entweder die angeforderten Daten oder Statusinformationen oder einen SOAP_FAULT. Dieses Verhalten ermöglicht im .NET Framework die Nutzung des SoapFaultError. Mit dem Try-Catch-Mechanismus lässt sich damit ein komfortables Fehlerhandling realisieren.

3.7.4 Prinzipielles zu Subscriptions

Einleitung

Um den Status eines Traces abzufragen, ist "GetStatus" aufzurufen. Um möglichst schnell von einer Statusänderung zu erfahren, ist ein sehr hochfrequentes Pollen nötig, welches in der Steuerung viel unnötige CPU-Belastung erfordert und im Netzwerk viel Traffic verursacht.

Um diesen Vorgang zu optimieren, bietet OPC XML-DA so genannte Subscriptions an. Hier wird so lange keine Antwort auf eine Anfrage geschickt, bis sich eine gewünschte Variable ändert oder ein Timeout auftritt. Die Verbindung wird also offen gehalten, ohne dass Traffic entsteht. Sobald für den Client relevante Daten vorhanden sind, werden ihm diese geschickt.

Diesen Mechanismus kennt der Trace via SOAP Webservice auch. Allerdings wird hierbei nur auf den Status des Traceobjekts angefragt, da dies die einzige wertvolle Information in diesem Umfeld ist.

Sobald sich der Status ändert (z. B. RUNNING -> FINISHED) werden die Clients, die vorher die Anfrage gestellt haben, eine entsprechende Antwort enthalten. Im Wesentlichen sind beliebig viele Clients möglich (solange die Ressourcen ausreichen).

Ablauf

Der Ablauf einer Subscription sieht wie folgt aus:

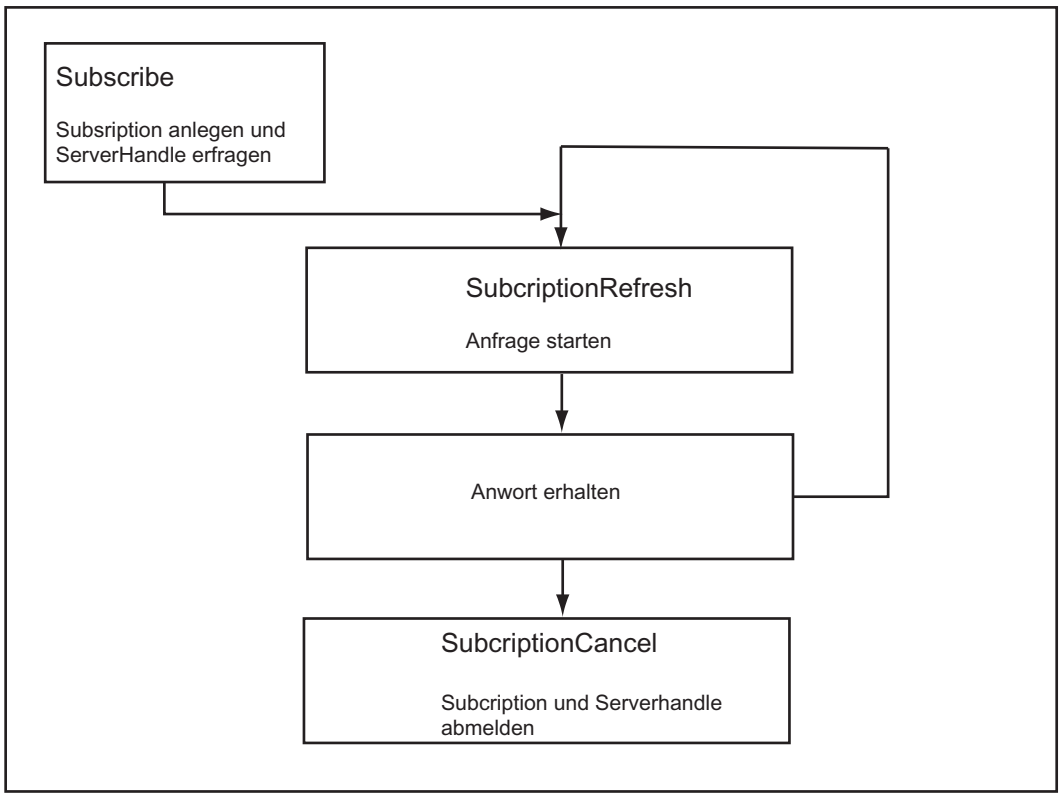


Bild 3-87 Subscription

Zunächst muss eine Subscription angelegt werden. Diese wird mit einem eindeutigen ServerHandle beantwortet, welche für die weitere Kommunikation benötigt wird.

Um eine neue Anfrage zu starten, kann SubscriptionRefresh beliebig oft aufgerufen werden. Als Parameter erhält dieser Request zwei Zeitangaben in Millisekunden:

- HoldTime:
  Diese Zeit sagt aus, wie lange mindestens mit der Beantwortung gewartet werden soll, egal, ob sich der Zustand geändert hat oder nicht.
- WaitTime:
  Nach Ablauf der HoldTime beginnt die WaitTime. Hat sich der Trace Status geändert erfolgt die Rückantwort des aktuellen Zustandes sofort. Ohne Änderung erfolgt die Rückantwort nach Ablauf der WaitTime.

Die genauen Methodenaufrufe werden im nächsten Abschnitt erklärt.

3.7.5 Schnittstelle

3.7.5.1 Globale Definitionen

TraceStateEnum

Enumerator, der den Status des Traceobjekts angibt.

Deklaration:

```
public enum TraceStateEnum
{
  RUNNING,
  STOPPED,
  ERROR,
  EMPTY,
  FINISHED,
  TRIGGERED
}
```

TraceDataCycleEnum

Enumerator, der bestimmt, in welchem Takt die Daten aufzuzeichnen sind. Hierbei ist auch zu beachten, dass große Traces u. U. zu einem Ebenenüberlauf führen können!

Deklaration:

```
public enum TraceDataCycleEnum
{
  IPO1,
  IPO2,
  SERVO,
}
```

Struktur VDSC

Struktur, die Informationen über die getraceten Variablen enthält. Dies sind:

- Variablenname `VarName`
- Variablentyp `VarType` in S7-Notation (z. B. DINT oder BYTE)
- `VarOffset` legt den Offset der Variablen innerhalb des Datenstroms (relativ zum Anfang des I/O-Containers) fest
- Variablenlänge `VarLen` (optional)

Deklaration:

```
public class VDSC
{
  public string VarName;
  public string VarType;
  public System.UInt32 VarOffset;
  public System.UInt32 VarLen;
}
```

TriggerCondition

Struktur, die den Trigger eines Traces angibt. Sie enthält die zu vergleichende Variable in symbolischen Namen gemäß VarProvider Notation.

- `Variable` bezeichnet die Variable, die verglichen werden soll.
- `Consant1` ist eine Konstante, die abhängig von der Triggerart gesetzt wird.
- `Constant2` ist eine Konstante, die abhängig von der Triggerart gesetzt wird.
- `Operation` gibt den Vergleichstypus als Enumerator `TriggerOperationType` an.
- `MatchCountTriggerPoint` gibt an, wie oft die Triggerbedingung zutreffen soll, bevor der Trigger aktiviert wird.
- `GlobalTriggerID` ist optional und enthält die eine eindeutige ID, die beim Einrichten eines verteilten Trace vom Browser erzeugt wird.

Deklaration:

```
public class TriggerCondition
{
  public string Variable;
  public string Consant1;
  public string Constant2;
  public TriggerOperationType Operation;
  public System.UInt32 MatchCountTriggerPoint;
  public System.UInt32 GlobalTriggerID;
}
```

Dazu den Vergleichstypus:

Aufruf:

```
public enum TriggerOperationType {
 RE,
 FE,
 WIB,
 OOB,
 BHV
 }
```

Ein Überblick über verschiedenen möglichen Kombinationen von Triggerart und Konstanten gibt die folgende Tabelle.

Triggerart	Symbol	Erklärung	Constant1	Constant2
Rising Edge	RE	Löst aus, wenn Variable den Wert von Constant1 in positiver Richtung überschreitet	Grenze, die überschritten wird	-nicht verwendet-
Falling Edge	FE	Löst aus, wenn Variable den Wert von Constant1 in negativer Richtung überschreitet	Grenze, die überschritten wird	-nicht verwendet-
Within a tolerance Band	WIB	Löst aus, wenn sich Variable innerhalb des von Constant1 und Constant2 aufgespannten Intervalls befindet	untere Grenze	obere Grenze
Out of tolerance Band	OOB	Löst aus, wenn sich Variable außerhalb des von Constant1 und Constant2 aufgespannten Intervalls befindet	untere Grenze	obere Grenze
Bit pattern	BHV	Löst aus, wenn Variable logisch verundet mit Constant1 gleich Constant2 ergibt, ((v&c1)==c2)	Bitmaske	Vergleichsergebnis

Übersicht der Triggerarten und Konstanten

Enumerator, der die Art des Traces bestimmt.

Aufruf:

```
public enum TraceStartTypeEnum
{
  IMMEDIATE,
  ENDLESS,
  TRIGGERED,
}
```

3.7.5.2 Methoden

StartTrace

Die Methode `StartTrace` startet einen Initialisierten Trace. Es wird dem SoapFault "No Trace available" zurückgeben, wenn noch kein Trace initialisiert wurde. `StartTrace` wird ignoriert (mit positivem Ergebnis), falls der Trace bereits läuft.

Die `GlobalTriggerID` muss bei einem verteilten Trace übergeben werden. Die Trigger-ID ist eine eindeutige ID, die die aktiv triggernde Station eindeutig identifiziert und für alle Stationen gleich ist.

Beim Neuanlauf eines Trace muss die Trigger-ID neu vergeben werden, um die Trigger-Ereignisse voneinander unterscheiden zu können.

Aufruf:

```
public TVS_Client.TVS.StartTrace_Response StartTrace
(
  uint GlobalTriggerID
)

public class StartTrace_Response
{
  public TraceStateEnum TraceState;
}
```

StopTrace

Die Methode `StopTrace` stoppt einen Laufenden Trace. Es wird dem SoapFault "No Trace available" zurückgeben, wenn noch kein Trace initialisiert wurde. Es wird ignoriert (mit positivem Ergebnis), falls der Trace bereits gestoppt ist.

Aufruf:

```
public TVS_Client1.TVS.StopTrace_Response StopTrace ( )

public class StopTrace_Response
{
  public TraceStateEnum TraceState;
}
```

CancelTrace

Die Methode `CancelTrace` löscht einen Aktiven Trace. Der Trace wechselt in den Zustand EMPTY, und alle Daten des Trace werden gelöscht. (Hinweis: Angeforderte, aber noch nicht abgeholte Datenblöcke des WebTrace werden auch gelöscht (siehe WebTrace::ReadData())

Es wird dem SoapFault "No Trace available" zurückgeben, wenn noch kein Trace initialisiert wurde.

Aufruf:

```
public TVS_Client.TVSIO.CancelTrace_Response CancelTrace ()

public class CancelTrace_Response
{
  public TraceStateEnum TraceState;
}
```

GetStatus

Die Methode `GetStatus` liefert den aktuellen Zustand des Trace. Wenn ein Traceobjekt gelöscht bzw. ungültig geworden ist, wird TraceIsValid "false" enthalten. Hierauf ist der Trace mittels `CancelTrace` zu löschen.

Aufruf:

```
public TVS_Client.TVSIO.GetStatus_Response GetStatus ( )

public class GetStatus_Response
{
  public bool TraceIsValid;
  public TraceStateEnum TraceState;
}
```

ReadData

Die Methode `readData` speichert Trace-Daten auf der RAM-Disk ab und liefert im Rückgabewert die URLs der Dateien. Diese Dateien können mit einem HTTP-GET-Request vom Client abgeholt werden.

Es wird der SoapFault "No Tracedata available" zurückgegeben, falls keine Trace-Daten vorhanden sind.

Aufruf:

```
public ReadData_Response ReadData()

public class ReadData_Response
{
  public TraceStateEnum TraceState;
  public string[] URL;
}
```

InitializeWebTrace

Mit `InitializeWebTrace` wird ein Trace angelegt. `VariablesToTrace` ist die Liste der symbolischen Namen gemäß VarProvider Notation. `TraceDataCycle` bestimmt den Takt, in dem die Daten aufgezeichnet werden sollen. `TraceStartType` bestimmt die Art des Traces. `Duration` gibt die Dauer der Aufzeichnung in Millisekunden an. Bei einem Endlostrace gibt dieser Parameter die Größe des Ringpuffers in Millisekunden an.

`MatchCountTriggerPoint` in `TriggerInformation` bestimmt, wie oft der Trigger auftreten muss, bevor getriggert und damit mit der Aufzeichnung begonnen wird. `Pretrigger` gibt die Anzahl der Werte an, die vor der Triggerung aufzunehmen sind ("Historie").

Aufruf:

```
public InitializeWebTrace_Response InitializeWebTrace
(
  string[] VariablesToTrace,
  TraceDataCycleEnum TraceDataCycle,
  TraceStartTypeEnum TraceStartType,
  uint Pretrigger,
  uint Duration,
  TriggerCondition TriggerInformation
  string[] DevicesInvolved
)
```

```
public class InitializeWebTrace_Response
{
  public VDSC[] CurrentlyTracedVariables;
  public TraceStateEnum TraceState;
  public string UID;
  public string[]DevicesInvolved;
}
```

InitializeWebTraceEx

InitializeWebTraceEx ist mit InitializeWebTrace identisch bis auf den Rückgabewert, in dem die Variablen nach ihrem Offset sortiert sind.

GetTraceParameters

Mit `GetTraceParameters` kann eine bestehende Trace-Projektierung ausgelesen werden.

Zurückgegeben wird nur ein eventuell bestehender WebTrace.

Aufruf:

```
public GetTraceParameters_Response GetTraceParameters
(
  string UID
)
```

```
public class GetTraceParameters_Response
{
  public TraceTypeEnum TraceType;
  public VDSC[] CurrentlyTracedVariables;
  public TraceStateEnum TraceState;
```

```
  public TraceDataCycleEnum TraceDataCycle;
  public string UID;
  public TraceStartTypeEnum TraceStartType;
  public unsignedInt Pretrigger;
  public unsignedInt Duration;
  public TriggerCondition TriggerInformation;
  public unsignedInt IOContainerOffset;
  public unsignedInt IOContainerLength;
  public hexBinary ClientHandle;
  public string[]DevicesInvolved;
}
```

EnableTrigger

Nur für den verteilten Trace.

Nachdem ein verteilter Trace eingerichtet und gestartet wurde, werden hiermit die Trigger aktiviert. Die `TriggerID` muss eindeutig sein, damit die Stationen diese unterscheiden können und nicht im Netz zirkulieren. Die Reihenfolge ist wichtig: da erst alle Traces laufen sollten, bevor die Trigger aktiviert werden, um den Verlust von Triggerereignissen zu vermeiden.

Rückgabe: TriggerState aktiviert oder nicht aktiviert.

Aufruf:

```
public EnableTrigger_Response EnableTrigger
(
    uint TriggerID
)
```

```
public class EnableTrigger_Response
{
  public TraceStateEnum TraceState;
}
```

ReadData

Mit ReadData wird der TVS-Service aufgefordert, den Puffer des Traces auszulesen und die Daten in temporäre Dateien zu verpacken. Diese sind unter den in URL angegebenen relativen Pfaden dann per HTTP abrufbar. Sollte der Puffer leer sein, wird die Anfrage mit dem SoapFault "No Tracedata available" beantwortet. Derzeit werden maximal 8 zusammengefasste Dateien mit maximal 8192 Aufzeichnungspunkten pro Anfrage geliefert.

ReadDataArchive

Ist ein Trace im Zustand STOPPED, können mit dieser Funktion die aufgezeichneten Daten angefordert werden.

Die Funktion liefert eine URL, unter der ein WTRC-Datenarchiv heruntergeladen werden kann, welches anschließend im WebTraceViewer angezeigt werden kann.

Hinweis

Die WTRC-Datei wird gelöscht, sobald sie einmal heruntergeladen wurde.

Aufruf:

```
public ReadDataArchive_Response ReadDataArchive ()
```

```
public class ReadDataArchive_Response
{
  public TraceStateEnum TraceState;
  public string URL
}
```

ReadDataArchives

Nur für den verteilten Trace.

ReadDataArchives holt selbstständig bei allen an einem verteilten Trace beteiligten Stationen die Tracedaten ab und bündelt sie in einer WTRC-Datei. Über das Array `URLField` wird eine Liste mit den URLs aller am Trace beteiligten Stationen übergeben. Der Trace muss zur Ausführung dieser Methode im Zustand STOPPED sein.

Die Rückgabe ist identisch mit der von ReadDataArchive.

Aufruf:

```
public ReadDataArchives_Response ReadDataArchives
(
  public string[] URLField;
)
```

```
public class ReadDataArchives_Response
{
  public TraceStateEnum TraceState;
  public string URL
}
```

3.7.5.3 Subscriptions

Einleitung

Nachfolgend sind Methoden für Subscription aufgeführt.

Subscribe

Mit der Methode `Subscribe` wird eine Subscription angelegt. Als Antwort kommt ein `ServerHandle`, mit dem sich ein Subscription-Vorgang eindeutig identifizieren lässt. Außerdem wird der aktuelle Trace Status mitgeliefert.

Aufruf:

```
public TVS_Client.TVS.Subscribe_Response Subscribe ( )

public class Subscribe_Response {
  public System.UInt32 ServerHandle;
  public TraceStateEnum TraceState;
}
```

SubscriptionRefresh

Mit SubscriptionRefresh wird erneut der Status des Trace abgefragt, welcher zuvor mit der Methode Subscribe erstmalig abgefragt wurde. Die Antwort des Servers kommt entweder HoldTime (Millisekunden) + WaitTime (Millisekunden), wenn sich der Status während dieser Zeit nicht geändert hat. Oder die Antwort kommt (frühestens) nach Ablauf der HoldTime und vor Ablauf der WaitTime, wenn sich während der WaitTime der Status des Traces ändert. Die HoldTime wird also auf jeden Fall gewartet.

In der Antwort sagt StateChanged aus, ob sich der Status zwischen Anfrage und Antwort geändert hat (true) oder ob der Status TraceState dem Status bei der Anfrage entspricht (false = Waittime abgelaufen).

Aufruf:

```
public TVS_Client.TVS.SubscriptionRefresh_Response SubscriptionRefresh (
  System.UInt32 ServerHandle ,
  System.UInt32 WaitTime ,
  System.UInt32 HoldTime
)

public class SubscriptionRefresh_Response {
 public bool StateChanged;
 public TraceStateEnum TraceState;
}
```

SubscriptionCancel

Mit `SubscriptionCancel` wird eine Subscription beendet und die Ressource freigegeben. Die Antwort gibt an, ob der Cancel erfolgreich war. Eventuell laufende SubscriptionRefreshs werden abgebrochen und sofort beantwortet.

Aufruf:

```
public TVS_Client.TVS.SubscriptionCancel_Response SubscriptionCancel (
  System.UInt32 ServerHandle )

public class SubscriptionCancel_Response {
  public bool SubscriptionCanceled;
 }
```

3.8 Variablen Provider

3.8.1 Überblick

Variablen Provider

Die Daten des SIMOTION Geräts sind über "Variablen Provider" erreichbar. Jeder Provider ermöglicht den Zugriff auf bestimmte Variablen.

Zurzeit existieren vier Variablen Provider, die im Folgenden beschrieben werden.

- MinWeb
- SIMOTION
- SIMOTION diagnostics
- UserConfig

Auf die durch die Variablen Provider bereitgestellten Daten können Sie über SIMOTION IT OPC XML-DA, über die SIMOTION IT DIAG (Diagnose-Standardseiten) und bei Bedarf über anwenderdefinierte HTML Seiten zugreifen.

3.8.2 SIMOTION

Über den Provider "SIMOTION" ist der Zugriff auf SIMOTION Prozessvariablen möglich. Ab V4.1 kann auch der Betriebszustand geändert, Sicherungen mit RamToRom und ActiveToRam angestoßen sowie auf Antriebsparameter und technologische Alarme zugegriffen werden.

Hinweis

Eine exakte Liste mit Beschreibung ist der Onlinehilfe des SIMOTION SCOUT Kapitel "Systemfunktionen, Systemvariablen und Konfigurationsdaten" zu entnehmen.

Variablensyntax des Providers "SIMOTION"

Der Zugriff auf Variablen des SIMOTION Geräts erfolgt bei OPC XML-DA V1.0 über die Begriffe "ItemPath" und "ItemName". In anwenderdefinierten HTML Seiten erfolgt der Zugriff über den "ItemName".

ItemPath

Der Name für "ItemPath" ist für SIMOTION Prozess Variablen immer "SIMOTION".

ItemPath="SIMOTION"

Hinweis

Der "ItemPath" wird ausschließlich für den Zugriff über OPC XML-DA benötigt, nicht für SIMOTION IT DIAG und in anwenderdefinierten HTML Seiten.

3.8.2.1 Zugriff auf Systemvariablen / TO-Systemvariablen

Für **Systemvariablen** lautet die **ItemName**-Syntax:

ItemName="var/name"

Beispiel: ItemName="var/userData.user3"

Für **TO-Systemvariablen** lautet die **ItemName**-Syntax:

ItemName="to/name.variable"

Beispiel: ItemName="to/Achse_1.positioningState.actualPosition"

Hinweis

Die zu verwendenden Namen der Systemvariablen und TO-Systemvariablen sind der Onlinehilfe des SIMOTION SCOUT Kapitel "Systemfunktionen, Systemvariablen und Konfigurationsdaten" zu entnehmen.

Für Unit-Variablen im Interface lautet die **ItemName**-Syntax:

ItemName=" unit/name.variable"

Beispiel: ItemName=" unit/prog_1.var_1"

Hinweis

Die zu verwendenden Namen für die Unit-Variablen im Interface entsprechen den Programm- und Variablennamen **in Kleinschreibung**.

3.8.2.2 Zugriff auf TO-Konfigurationsdaten (ab V4.1)

Für **TO-Konfigurationsdaten** lautet die **ItemName**-Syntax:

ItemName="cfg/TOName.activeConfigData|setConfigData.variable"

activeConfigData:	aktuell gültige Konfigurationsdaten, nur lesbar
setConfigData:	Datensatz-Abbild, Schreibzugriff möglich Das Schreiben der Daten ist möglich, wenn die Eigenschaft "effectivenes" den Wert "CHANGEABLE_WITH_RESTART" oder "CHANGEABLE_WITHOUT_RESTART" enthält. Im Falle von "CHANGEABLE_WITH_RESTART" wird die Änderung erst nach einem Restart des entsprechenden TOs gültig.

Beispiel: ItemName="cfg/Achse_0.setConfigData.Restart.restartActivationSetting"

Hinweis

Die zu verwendenden Namen der TO-Konfigurationsdaten sind der Onlinehilfe des SIMOTION SCOUT Kapitel "Systemfunktionen, Systemvariablen und Konfigurationsdaten" zu entnehmen.

3.8.2.3 Zugriff auf Antriebsparameter (ab V4.1)

Für **Antriebsparameter** lautet die **ItemName**-Syntax:

ItemName="drv/TOName|LogAddr.Params.ParamNo"

TOName:	Angabe des TO-Namen (möglich, wenn für das Antriebsobjekt ein Achs-TO existiert)
LogAddr:	Angabe der logischen Antriebsadresse
ParamNo:	Parameternummer Bei einem Schreibzugriff auf eine nicht schreibbare Antriebsvariable erfolgt eine entsprechende Rückmeldung (Fehlercode) vom Antrieb.

Beispiel 1: ItemName="drv/Achse_0.Params.105"

Beispiel 2: ItemName="drv/256.Params.5"

3.8.2.4 Zugriff auf technologische Alarme (ab V4.1)

Für **technologische Alarme** lautet die **ItemName**-Syntax:

ItemName="dev/Alarm.Variable|Values-Array

Variable:
- State
  Status der Abfrage:
  READY
  BUSY
  ERROR
- Version
  Wird bei jeder Änderung des Alarmpuffers inkrementiert. Durch den Eintrag dieser Variable in eine Subscription kann man sich von einer Änderung des Alarmpuffers benachrichtigen lassen.
- EventCount
  Anzahl aktuell anstehender Alarme
- QuitAll
  Quittieren aller anstehender Alarme

Values-Array: Array mit den aktuell anstehenden Alarmen

Das Array enthält so viele Elemente, wie in EventCount eingetragen sind.

Beispiel: ItemName="dev/Alarm.Version"

Für einen aktuell anstehenden Alarm lautet die **ItemName**-Syntax:

ItemName="dev/Alarm.Values[ValueNumber].ArrayElement"

ValueNumber: Index eines Alarms in der Liste der aktuell anstehenden technologischen Alarme

ArrayElement:
- AlarmNo
  Alarmnummer
- To
  Name des TOs, das den Alarm erzeugt hat
- Time
  Zeitpunkt des Alarmeintrags
- Text
  Alarmtext
- Quit
  Quittieren des Alarms
- Type
  Klassifizierung des technologischen Alarms:
  ALARM
  WARNING
  INFORMATION

Beispiel: ItemName="dev/Alarm.Values[0].AlarmNo"

3.8.2.5 Betriebszustand ändern (ab V4.1)

Für die Einstellung des Betriebszustandes lautet die **ItemName**-Syntax:

ItemName="dev/Service.BZU.Variable"

Variable:
- Value
  Durch Schreiben eines der folgenden Werte wird der Betriebszustand entsprechend geändert:
  - STOP
  - STOPU
  - RUN
- State
  Anzeige der Ausführungsstati bei Betriebszustandsänderung
  Dabei wechseln die Stati von IDLE über ACTIVE nach READY.
- Result
  Ergebnis der Betriebszustandsänderung (wenn State = READY)
  Wenn der Betriebszustand erfolgreich geändert wurde, ist Result = OK. Andernfalls ist Result = Fehlerkennung

Beispiel: ItemName="dev/Service.BZU.Value"

3.8.2.6 RamToRom (ab V4.1)

Für das Ausführen von **RamToRom** lautet die **ItemName**-Syntax:

ItemName="dev/Service.RamToRom.Variable"

Variable:
- Value
  Abspeichern starten mit Value = 0
- State
  Statusanzeige des Speichervorgangs
  Die Anzeige zählt von 0% bis 100% hoch.
- Result
  Ergebnis des Speichervorgangs (wenn State = 100%)
  Wenn der Speichervorgang erfolgreich beendet wurde, ist Result = OK. Andernfalls ist Result = Fehlerkennung

Beispiel: ItemName=" dev/Service.RamToRom.Value"

3.8.2.7 ActiveToRam (ab V4.1)

Für das Ausführen von **ActiveToRam** (nach Änderung von Konfigurationsdaten) lautet die **ItemName**-Syntax:

ItemName="dev/Service.ActToRam.Variable"

Variable:
- Value
  Abspeichern starten mit Value = 0
- State
  Statusanzeige des Speichervorgangs
  Die Anzeige zählt von 0% bis 100% hoch.
- Result
  Ergebnis des Speichervorgangs (wenn State = 100%)
  Wenn der Speichervorgang erfolgreich beendet wurde, ist Result = OK. Andernfalls ist Result = Fehlerkennung

Beispiel: ItemName=" dev/Service.ActToRam.Value"

3.8.2.8 Zugriff auf die globalen Variablen (ab V4.2)

Der Zugriff auf die vom Anwender im SCOUT angelegten "Geräteglobalen Variablen" der Steuerung geschieht über /glo/.

Für die **Geräteglobalen Variablen** lautet die **ItemName**-Syntax:

ItemName="glo/name"

Damit diese Variablen sichtbar werden, muss die Symbolinformation in die Steuerung geladen werden. Dazu muss im SCOUT unter **Gerät > Eigenschaften > Einstellungen** ein Häkchen gesetzt werden.

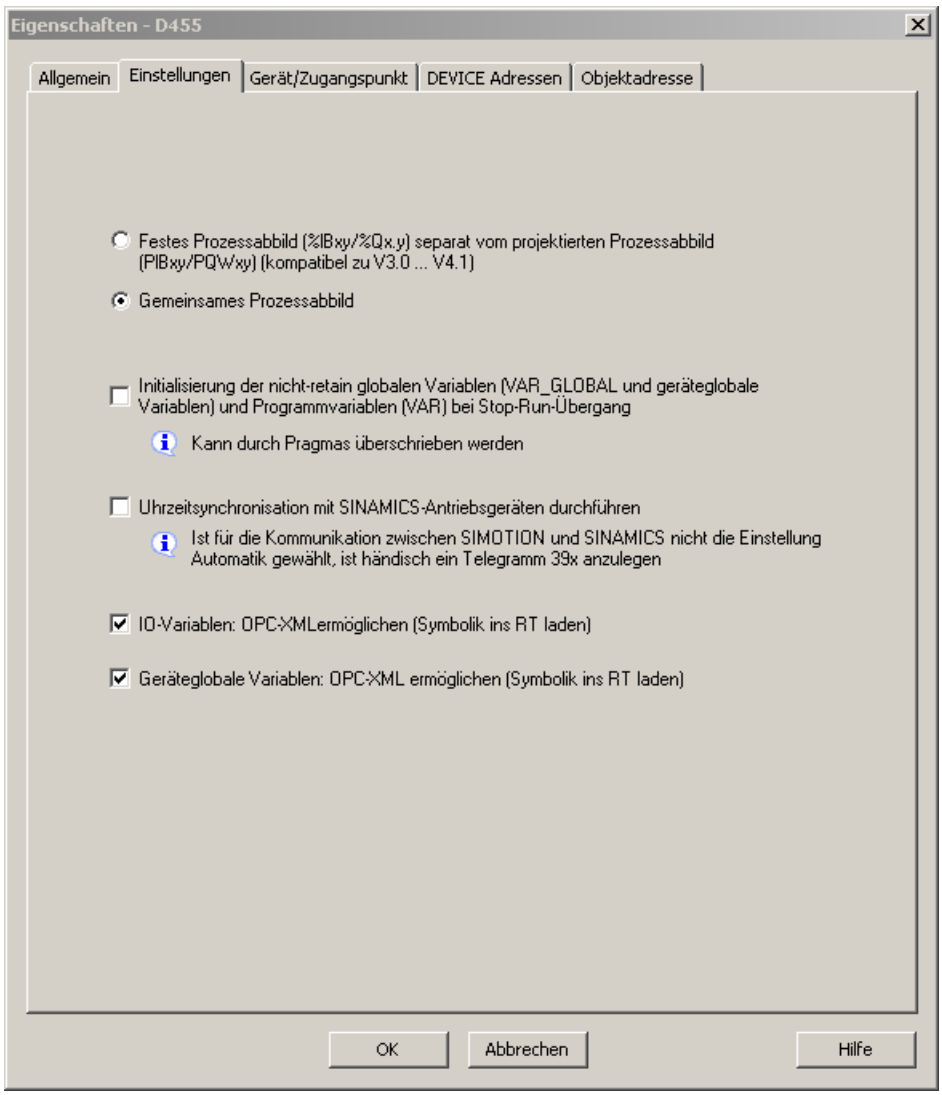


Bild 3-88 Einstellung der 'Geräteglobale Variablen' in den Geräteeigenschaften des SCOUT

3.8.2.9 Zugriff auf die IO Variablen (ab V4.2)

Der Zugriff auf die Adressliste, der im SCOUT angelegten I/O Variablen der Steuerung geschieht auf 3 verschiedene Arten:

- /io/_direct/
  Adressiert den Peripheriedirektzugriff (Aktualwerte) zu den I/O-Variablen.
  Dieser Zugriff wird für alle I/O Variablen angeboten.
- /io/_image/
  Adressiert das Prozessabbild zu den I/O-Variablen.
  Angezeigt werden nur die I/O-Variablen im Adressbereich 0-63, die Adressen der Form PI... / PQ... haben. I/O-Symbole der Form %I... / %Q... sind derzeit nicht adressierbar.
- /io/_quality/
  Adressiert die Quality von I/O-Variablen, d. h. den I/O-Status des Subslot (aus der HW-Konfig) der diese I/O-Variable beinhaltet.
  Dies ist ein Bitmuster von 32 Bit. Im Handbuch 'SIMOTION ST Structured Text' befindet sich im Abschnitt ' Zugriff auf die IO Variablen (ab V4.2) ' eine Übersicht der möglichen Werte des Bitmusters.
  Die Quality ist für alle I/O-Variablen in einem Subslot gleich. Die Quality wird für die einzelnen I/O-Variablen der Grunddatentypen (BIT, BYTE, WORD, DWORD) und für Arrays als Ganzes angeboten. Sie wird nicht angeboten für Array-Elemente (d.h., Arrays sind nicht aufklappbar).

Für die IO **Variablen** lautet die **ItemName**-Syntax:

ItemName="io/_direct|_image|_quality/name"

Damit diese Variablen sichtbar werden, muss die Symbolinformation in die Steuerung geladen werden. Dazu muss im SCOUT unter **Gerät > Eigenschaften > Einstellungen** ein Häkchen gesetzt werden.

3.8.2.10 Zugriff auf die AlarmS-Medlungen (ab V4.2)

Der Zugriff auf die vom Anwender im SCOUT angelegten und von der Steuerung ausgelösten AlarmS-Meldungen.

Für die **AlarmS-Meldungen** lautet die **ItemName**-Syntax:

ItemName="dev/AlarmS.Values[ValueNumber].ArrayElement"

ValueNumber: Index eines AlarmS in der Liste der aktuell anstehenden technologischen Alarme

ArrayElement:
- AlarmNo
  Alarmnummer
- AddInfo
  Zusatzinfos
- Time
  Zeitpunkt des AlarmS-Eintrags
- Text
  AlarmS-Text
- Quit
  Quittieren des AlarmS
- Type
  S / SQ

Beispiel: ItemName="dev/AlarmS.Values[0].AlarmNo"

3.8.3 SIMOTION diagnostics

3.8.3.1 Einleitung

Zugriff auf Diagnosevariablen

Über den Provider "SIMOTION diagnostics" kann auf die Diagnosevariablen einer SIMOTION Steuerung zugegriffen werden.

Auf die meisten Variablen kann nur lesend und auf einige wenige (z. B. Betriebszustand) kann auch schreibend zugegriffen werden. Alle Variablen sind vom Typ String. Zahlenwerte werden also vom Provider in Strings umgewandelt.

Der Variablenhaushalt ist dynamisch und hängt von der aktuellen Konfiguration der SIMOTION Steuerung ab. Der Provider unterstützt das Browsen via OPC XML-DA V1.0, sodass der aktuelle Variablenhaushalt über Browsing in Erfahrung gebracht werden kann.

Variablengruppen des Providers "SIMOTION diagnostics"

Die Diagnosevariablen des Providers "SIMOTION diagnostics" sind in Gruppen zusammengefasst.

Der Name einer Variable ergibt sich dann aus Gruppenname und Variablenname:

Z. B.: Gruppe.Variable

3.8.3.2 Gruppe DeviceInfo

Allgemeine Informationen zum SIMOTION Gerät

Die Gruppe DeviceInfo enthält allgemeine Informationen zum SIMOTION Gerät. Die 10 Variablen dieser Gruppe sind immer verfügbar.

Tabelle 3- 16 Variablen der Gruppe DeviceInfo

Variable	Beschreibung
DeviceInfo.Board	Gibt an, um welches System es sich handelt, nur lesend
DeviceInfo.Licence-Serial-Nr	Lizenz Serien Nummer für dieses Device, nur lesend
DeviceInfo.BZU	Zugriff auf den Betriebszustand, lesend und schreibend, gültige Werte zum Schreiben: STOP, STOPU, RUN
DeviceInfo.Systemtime	Zugriff auf die Systemzeit, lesend und schreibend, die Zeit muss immer wie in folgendem Beispiel angegeben werden: "Tue Aug 05 17:00:00 2003", ein anderes Format wird nicht akzeptiert.
DeviceInfo.Timezone	Zeitverschiebung in Minuten, lesend und schreibend, gültige Werte sind -720 bis +720
DeviceInfo.Active-MAC	Active MAC Adresse, nur lesend
DeviceInfo.Remanent-MAC	Remanente MAC Adresse, nur lesend
DeviceInfo.IP-Address	IP Konfigurationsdaten (Adresse, Subnetmask und Gateway), nur lesend
DeviceInfo.Subnet-Mask	
DeviceInfo.Gateway	

Weitere Variablen der Gruppe DeviceInfo

Die folgenden Variablen liefern HTML Farbwerte ("#XXXXXX") die den Farben der LEDs DC5V, RUN, STOPU und STOP des SIMOTION Geräts entsprechen. Damit ist es z. B. möglich, über eine HTML-Tabelle (mittels des Attributes "background" in den Zellen) den Betriebszustand als "Ampelinfo" darzustellen, ähnlich der Anzeige im SIMOTION SCOUT, wie bei "Betriebszustand ..."

Auf diese Werte kann nur lesend zugegriffen werden.

Tabelle 3- 17 Variablen der Gruppe DeviceInfo

Variable	Beschreibung
DeviceInfo.LEDColor.DC5V	Farbe für LED DC5V, da der Server nur ansprechbar ist, wenn Spannung anliegt, ist die zugehörige HTML Farbe immer grün ("#00FF00")
DeviceInfo.LEDColor.RUN	Farbe für LED RUN, im Betriebszustand RUN grün ("#00FF00") sonst grau ("#C0C0C0")
DeviceInfo.LEDColor.STOPU	Farbe für LED STOPU, im Betriebszustand STOPU orange ("#FF9900") sonst grau ("#C0C0C0")
DeviceInfo.LEDColor.STOP	Farbe für LED STOP, im Betriebszustand STOP orange ("#FF9900") sonst grau ("#C0C0C0")

3.8.3.3 Gruppe CompInfo

Diese Gruppe liefert Informationen zu den Komponenten des Geräts. In dieser Gruppe variiert der Variablenhaushalt je nach Anzahl der Technologie Pakete oder der zusätzlichen Hardware.

Auf alle Variablen kann nur lesend zugegriffen werden.

Informationen zur CPU

Folgende Variablen liefern Informationen zur CPU:

Tabelle 3- 18 Variablen der Gruppe CompInfo

Variable	Beschreibung
CompInfo.Cpu.MLFB	CPU MLFB / Bestellnummer
CompInfo.Cpu.Serial-Nr	CPU Seriennummer
CompInfo.Cpu.Revision-Nr	Revisionsnummer
CompInfo.Cpu.Kernelname	Kernelname
CompInfo.Cpu.Build-Nr	Build Nummer
CompInfo.Cpu.User-Version	User Version (Firmware)

Informationen zu den Technologie Paketen (TP) und zur Hardware

Mit den folgenden Variablen kann die Anzahl der vorhandenen TPs bzw. Hardware festgestellt werden.

Tabelle 3- 19 Variablen der Gruppe CompInfo

Variable	Beschreibung
CompInfo.TP-Count	Anzahl der vorhandenen Technologie Pakete
CompInfo.HW-Count	Anzahl der vorhandenen Hardware

Sind TPs vorhanden, so können mit CompInfo.TPx. Variablenname (wobei x für die TP Nummer steht) Informationen über die einzelnen TPs abgefragt werden.

Das erste TP erhält die Nummer 1 (nicht 0), zum Beispiel: CompInfo.TP1.Name

Folgende Informationen stehen zur Verfügung:

Tabelle 3- 20 Variablen der Gruppe CompInfo

Variable	Beschreibung
CompInfo.TPx.Name	Name des TP
CompInfo.TPx.User-Version	User Version des TP
CompInfo.TPx.Build-Nr	Build Nummer des TP

Sind zusätzliche Hardwarekomponenten vorhanden, so können mit CompInfo.HWx.Variablenname (wobei x für die Nummer der Hardware steht) Informationen über die einzelnen Hardwarekomponenten abgefragt werden.

Die erste Hardwarekomponente erhält die Nummer 1 (nicht 0), zum Beispiel: CompInfo.HW1.MLFB

Folgende Informationen stehen zur Verfügung:

Tabelle 3- 21 Variablen der Gruppe CompInfo

Variable	Beschreibung
CompInfo.HWx.MLFB	MLFB / Bestellnummer
CompInfo.HWx.Serial-Nr	Seriennummer
CompInfo.HWx.Revision-Nr	Revisionsnummer
CompInfo.HWx.Firmwarename	Firmware Name
CompInfo.HWx.Build-Nr	Build Nummer
CompInfo.HWx.User-Version	User Version

Da die Informationen dynamisch sind und der Umfang vorher nicht bekannt ist, existieren zur Vereinfachung der Anzeige von Hardwarekomponenten und TPs in HTML noch folgende Variablen:

Tabelle 3- 22 Variablen der Gruppe CompInfo

Variable	Beschreibung
CompInfo.TableHead.TP	Liefert den Kopf einer HTML-Tabelle mit allen Informationen über die TPs, z. B. "<tr><th>TP-Name</th><th>User-Ver.</th> <th>Build-Nr.</th></tr>"
CompInfo.Table.TP	Liefert eine HTML - Tabelle mit allen Informationen über alle vorhandenen TPs
CompInfo.TableHead.HW	Liefert den Kopf einer HTML-Tabelle mit allen Informationen über die Hardwarekomponenten, z. B. " <tr><th>MLFB</th><th>Serial-Nr.</th> <th>Revision-Nr.</th><th>FW-Name</th> <th>User-Ver.</th><th>Build-Nr.</th></tr> "
CompInfo.Table.HW	Liefert eine HTML - Tabelle mit allen Informationen über alle vorhandenen Hardwarekomponenten

Hinweis

Der getrennte Zugriff auf Tabelle und Tabellenkopf ermöglicht eine getrennte Formatierung.

3.8.3.4 Gruppe CPULoad

Informationen zur Auslastung der CPU

Die Gruppe CPULoad liefert Informationen über die Auslastung der CPU. Alle Variablen können nur lesend zugegriffen werden.

Tabelle 3- 23 Variablen der Gruppe CPULoad

Variable	Beschreibung
CPULoad.Percent	CPU Auslastung in Prozent
CPULoad.Mintime	Minimale Laufzeit der BackgroundTask (Freier Zyklus) in ms mit 5 Nachkommastellen.
CPULoad.Acttime	Aktuelle Laufzeit der BackgroundTask (Freier Zyklus) in ms mit 5 Nachkommastellen.
CPULoad.Maxtime	Maximale Laufzeit der BackgroundTask (Freier Zyklus) in ms mit 5 Nachkommastellen.

3.8.3.5 Gruppe MemoryLoad

Informationen über die Speicherauslastung

Die Gruppe MemoryLoad liefert Informationen über die Auslastung der Speichermedien in Bytes bzw. prozentual. Auf alle Variablen kann nur lesend zugegriffen werden.

Tabelle 3- 24 Variablen der Gruppe MemoryLoad

Variable	Beschreibung
MemoryLoad.Flash-Size	Größe des Flash Speichers.
MemoryLoad.Flash-Used	Aktuell belegter Flash Speicher.
MemoryLoad.RAM-Size	Größe des RAM.
MemoryLoad.RAM-Used	Aktuell belegter RAM.
MemoryLoad.RAMDisk-Size	Größe der RAM-Disk..
MemoryLoad.RAMDisk-Used	Aktuell belegter RAM-Disk Speicher.
MemoryLoad.Remanent-Size	Größe des remanenten Speichers.
MemoryLoad.Remanent-Used	Aktuell belegter remanenter Speicher.
MemoryLoad.Flash-Percent	Genutzter Anteil des externen Flash Speichers.
MemoryLoad.RAM-Percent	Genutzter Anteil des RAM Speichers.
MemoryLoad.RAMDisk-Percent	Genutzter Anteil der RAM Disk.
MemoryLoad.Remanent-Percent	Genutzter Anteil des internen Flash Speichers.

3.8.3.6 Gruppe TaskRT

Variablen der Gruppe TaskRT

Die Gruppe TaskRT liefert Informationen zu den Tasklaufzeiten und Taskzuständen des SIMOTION Geräts. Es werden die gleichen Werte geliefert wie im SIMOTION SCOUT unter Gerätediagnose, Tasklaufzeiten. Auf alle Werte kann nur lesend zugegriffen werden. Der Variablenhaushalt ist dynamisch und hängt von der Konfiguration des Ablaufsystems im SIMOTION SCOUT ab.

Tabelle 3- 25 Variablen der Gruppe TaskRT

Variable	Beschreibung
TaskRT.TaskCnt	Liefert die Anzahl der aktuell vorhandenen Tasks.

Tasknamen

Über TaskRT.Taskname.Variablenname können Informationen zu den einzelnen Tasks abgefragt werden. Die Tasks sind in IT DIAG und SCOUT gleich benannt.

Für jede Task können die gleichen Informationen abgefragt werden, hier am Beispiel der ersten MotionTask.

Beispiel:

TaskRT.MotionTask_1.Status

Aktueller Status der Task, kann eine sinnvolle Kombination aus folgenden Werten sein: STOP_PENDING, STOPPED, RUNNING, STOP_UNCOND, WAITING, SUSPENDED, WAITING_FOR_NEXT_CYCLE, WAITING_FOR_NEXT_INTERRUPT, LOCKED, SUSPENDED_BY_DEBUG_MODE

Weitere Variablen der Gruppe TaskRT

Tabelle 3- 26 Variablen der Gruppe TaskRT

Variable	Beschreibung
TaskRT.MotionTask_1.Actual	Aktuelle Laufzeit der Task in msec, mit 5 Nachkommastellen
TaskRT.MotionTask_1.Min	Minimale Laufzeit der Task in msec, mit 5 Nachkommastellen
TaskRT.MotionTask_1.Max	Maximale Laufzeit der Task in msec, mit 5 Nachkommastellen
TaskRT.MotionTask_1.Average	Mittlere Laufzeit der Task in msec, mit 5 Nachkommastellen

Da die Informationen dynamisch sind und der Umfang vorher nicht bekannt ist, existieren zur Vereinfachung der Anzeige der Taskinformationen in HTML noch folgende Variablen:

Tabelle 3- 27 Variablen der Gruppe TaskRT

Variable	Beschreibung
TaskRT.TableHead	Liefert den Kopf einer HTML-Tabelle mit allen Informationen Tasks, z. B. " <tr><th>Taskname</th><th>Status</th> <th>Actual</th><th>Min</th><th>Max</th> <th>Average</th></tr> "
TaskRT.Table	Liefert eine HTML-Tabelle mit allen Informationen über die vorhandenen Tasks, alle Laufzeitwerte werden mit Einheit eingetragen, die hier im Gegensatz zur Einzelwertabfrage zwischen s und ms variieren kann. Es werden 3 Stellen nach dem Komma angezeigt.

3.8.3.7 Gruppe DiagBuffer

Die Gruppe DiagBuffer liefert Informationen über die im DiagBuffer vorhandenen Ereignisse. Auf alle Variablen kann nur lesend zugegriffen werden.

Events können als Texte in Deutsch, Englisch, Französisch, Spanisch und Italienisch ausgegeben werden.

Voraussetzung

Voreingestellt ist die englische Textausgabe. Damit ein Event als Text in einer anderen Sprache angezeigt wird, muss eine Datei mit der gewünschten Sprache auf die Speicherkarte der SIMOTION Steuerung übertragen werden.

Sprache	Dateiname
Englisch	DGBUFTXT-EN.EDB
Deutsch	DGBUFTXT-DE.EDB
Französisch	DGBUFTXT-FR.EDB
Italienisch	DGBUFTXT-IT.EDB
Spanisch	DGBUFTXT-ES.EDB

Sprachspezifische Dateinamen der DiagBuffer Texte

Vorgehensweise

1. Öffnen Sie das Verzeichnis \3_Diag_Buf_Messages\Diag_Buf_Messages auf der SIMOTION IT DIAG DVD.
2. Legen Sie die Speicherkarte der SIMOTION Steuerung in ein Schreib-/Lesegerät ein.
3. Kopieren Sie die DGBUFTXT-XX.EDB Datei der gewünschten Sprache in das Verzeichnis \USER\SIMOTION\HMICFG. Sollte dieses Verzeichnis nicht existieren, erstellen Sie dieses.
4. Fügen Sie die Speicherkate wieder in das SIMOTION Gerät ein.

Vorgehensweise bei der SIMOTION P350

1. Beenden Sie die SIMOTION P Steuerung.
2. Öffnen Sie das Verzeichnis AddOn\4_Accessories\Simotion_IT\3_Diag_Buf_Messages\Diag_Buf_Messages auf der SIMOTION SCOUT Add-ons DVD.
3. Kopieren Sie die DGBUFTXT-XX.EDB Datei der gewünschten Sprache in das Verzeichnis F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG (bei einer Default-Installation).
4. Starten Sie die SIMOTION P Steuerung.

Vorgehensweise bei der SIMOTION P320

1. Beenden Sie die SIMOTION P Steuerung.
2. Öffnen Sie das Verzeichnis AddOn\4_Accessories\Simotion_IT\3_Diag_Buf_Messages\Diag_Buf_Messages auf der SIMOTION SCOUT Add-ons DVD.
3. Kopieren Sie die DGBUFTXT-XX.EDB Datei der gewünschten Sprache in das Verzeichnis D:\Card\USER\SIMOTION\HMICFG (bei einer Default-Installation).
4. Starten Sie die SIMOTION P Steuerung.

Hinweis

Es kann immer nur eine Sprache auf der SIMOTION Steuerung gespeichert werden.

Bei der Auslieferung und nach einem Firmware-Update befindet sich immer die englische Sprachversion auf dem Gerät.

Aus Kompatibilitätsgründen wird auch eine DGBUFTXT.EDB erkannt, wenn keine DGBUFTXT-XX.EDB Datei gefunden wird. Sind beide Dateien vorhanden, dann hat die DGBUFTXT-XX.EDB Vorrang.

Variablen der Gruppe DiagBuffer

Zur Vereinfachung der Anzeige stehen Ihnen folgende Variablen zur Verfügung:

Tabelle 3- 28 Variablen der Gruppe DiagBuffer

Variable	Beschreibung
DiagBuffer.TableHead	Liefert den Kopf einer HTML-Tabelle mit allen Ereignissen. Der Inhalt lautet: <tr><th>Nr</th><th>Time</th><th>Date</th><th>Event</th></tr>
DiagBuffer.Table	Liefert den Inhalt der HTML-Tabelle mit allen Ereignissen. Jede Zeile ist nach dem folgenden Format aufgebaut: <tr><td>NUMBER</td><td>TIME</td><td>DATE</td><td>EVENT</td></tr> **Hinweis:** Die im Format angegebenen Texte NUMBER, TIME, DATE und EVENT werden durch den entsprechenden Wert des jeweiligen Ereignisses ersetzt.
DiagBuffer.ExtendedTable	Liefert den Inhalt der HTML-Tabelle mit allen Ereignissen inklusive der erweiterten Einträge, die über den Info-Button angezeigt werden.
DiagBuffer.ExtendedBufferJScript	Liefert ein dynamisch erzeugtes JavaScript Fragment, das zur Darstellung der Tabelle benötigt wird.
DiagBuffer.LText[]	Liefert ein Array, das den Zugriff auf den kompletten Text des Diagnosepuffereintrags ermöglicht. Der Index entspricht dem Index des Diagnosepuffereintrags. Die einzelnen Elemente eines Diagnosepuffereintrags (Uhrzeit, Datum, Text, Text des erweiterten Eintrags) werden durch "/@@/" voneinander getrennt.

Über folgende Variablen können Sie auf die Daten von bestimmten Ereignissen im Diagnosepuffer direkt zugreifen:

Tabelle 3- 29 Variablen der Gruppe DiagBuffer - Direktzugriff

Variable	Beschreibung
DiagBuffer.EventCnt	Anzahl der aktuell im Diagnosepuffer vorhandenen Ereignisse
DiagBuffer.CplEventCnt	Ereigniszähler über die Umlaufpuffergrenze hinweg Der Zähler wird im Hochlauf mit der aktuellen Anzahl der Diagnosepuffereinträge initialisiert. Mit jedem weiteren Eintrag wird der Wert inkrementiert, auch über die maximale Anzahl Diagnosepuffereinträge hinweg.
DiagBuffer.Time_1 bis DiagBuffer.Time_n	Zeitpunkt des jeweiligen Ereignisses

Variable	Beschreibung
DiagBuffer.Date_1 bis DiagBuffer.Date_n	Datum des jeweiligen Ereignisses
DiagBuffer. Text_1 bis DiagBuffer.Text_n	Text des jeweiligen Ereignisses **Hinweis:** Sollten die jeweilige Ereignistextnummer und deren Parameter nicht aufgelöst werden können, so werden die Nummer und Parameter im HEX Format ausgegeben. Die Variable im HEX Format ist ein String bestehend aus 20 Hexadezimalzeichen (ohne Trennzeichen).

Beispiel für eine HTML-Seite

```
<html>
 <head>
  <title>SIMOTION <%=DeviceInfo.Board%> - Diagnostics</title>
  <script type="text/javascript">
   <%=DiagBuffer.ExtendedBufferJScript%>
  </script>
 </head>
 <body style="font-family: Arial">
  <h2>Diag Buffer (extended)</h2>
  <table border="2" cellspacing="1" cellpadding="5">
   <font size="4">
    <%=DiagBuffer.TableHead%>
    <%=DiagBuffer.ExtendedTable%>
   </font>
  </table>
 </body>
</html>
```

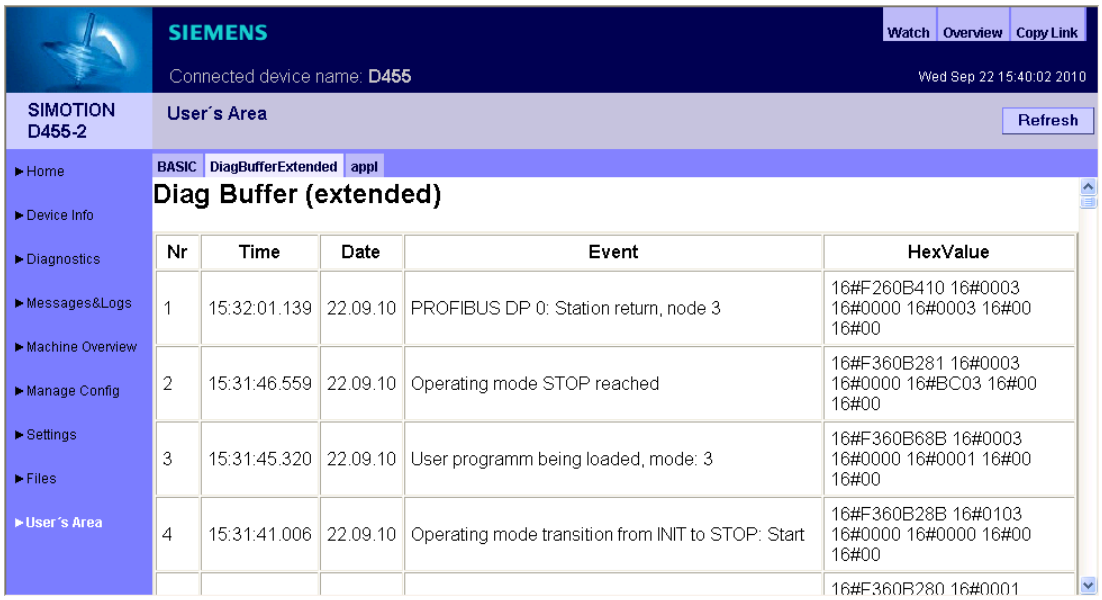


Bild 3-89 Ergebnis des Beispielcodes

3.8.3.8 Gruppe DiagBufferDrv

Die Gruppe DiagBufferDrv liefert Informationen über den Antriebsdiagnosepuffer. Auf alle Variablen kann nur lesend zugegriffen werden.

Variablen der Gruppe DiagBufferDrv

Variable	Beschreibung
DiagBufferDrv.TableHead	Liefert den Kopf einer HTML-Tabelle mit allen Ereignissen. Der Inhalt lautet: <tr><th>Nr</th><th>Time</th><th>Date</th><th>Event</th></tr>
DiagBufferDrv.Table	Liefert den Inhalt der HTML-Tabelle mit allen Ereignissen. Jede Zeile ist nach dem folgenden Format aufgebaut: <tr><td>NUMBER</td><td>TIME</td><td>DATE</td><td>EVENT</td></tr> **Hinweis:** Die im Format angegebenen Texte NUMBER, TIME, DATE und EVENT werden durch den entsprechenden Wert des jeweiligen Ereignisses ersetzt.
DiagBufferDrv.ExtendedTable	Liefert den Inhalt der HTML-Tabelle mit allen Ereignissen inklusive der erweiterten Einträge, die über den Info-Button angezeigt werden.
DiagBufferDrv.ExtendedBufferJScript	Liefert ein dynamisch erzeugtes JavaScript Fragment, das zur Darstellung der Tabelle benötigt wird.
DiagBufferDrv.LText[]	Liefert ein Array, das den Zugriff auf den kompletten Text des Diagnosepuffereintrags ermöglicht. Der Index entspricht dem Index des Diagnosepuffereintrags. Die einzelnen Elemente eines Diagnosepuffereintrags (Uhrzeit, Datum, Text, Text des erweiterten Eintrags) werden durch "/@@/" voneinander getrennt.

Über folgende Variablen können Sie auf die Daten von bestimmten Ereignissen im Antriebsdiagnosepuffer direkt zugreifen:

Tabelle 3- 30 Variablen der Gruppe DiagBufferDrv - Direktzugriff

Variable	Beschreibung
DiagBufferDrv.EventCnt	Anzahl der aktuell im Antriebsdiagnosepuffer vorhandenen Ereignisse
DiagBufferDrv.CplEventCnt	Ereigniszähler über die Umlaufpuffergrenze hinweg Der Zähler wird im Hochlauf mit der aktuellen Anzahl der Antriebsdiagnosepuffereinträge initialisiert. Mit jedem weiteren Eintrag wird der Wert inkrementiert, auch über die maximale Anzahl Antriebsdiagnosepuffereinträge hinweg.
DiagBufferDrv.Time[1] bis DiagBufferDrv.Time[n]	Zeitpunkt des jeweiligen Ereignisses

Variable	Beschreibung
DiagBufferDrv.Date[1] bis DiagBufferDrv.Date[n]	Datum des jeweiligen Ereignisses
DiagBufferDrv. Text[1] bis DiagBufferDrv.Text[n]	Text des jeweiligen Ereignisses **Hinweis:** Sollten die jeweilige Ereignistextnummer und deren Parameter nicht aufgelöst werden können, so werden die Nummer und Parameter im HEX Format ausgegeben. Die Variable im HEX Format ist ein String bestehend aus 20 Hexadezimalzeichen (ohne Trennzeichen).

3.8.3.9 Gruppe Alarms

Informationen zur Alarmtabelle

Die Gruppe Alarms liefert Informationen über die anstehenden Alarme. Auf alle Variablen kann nur lesend zugegriffen werden.

Tabelle 3- 31 Variablen der Gruppe Alarms

Variable	Beschreibung
Alarms.AlarmCnt	Anzahl der Alarme
Alarms.Table	HTML Tabelle mit allen anstehenden Alarmen
Alarms.TableHead	Tabellenkopf für die HTML-Tabelle der anstehenden Alarme
Alarms.TableHeadBuffer	HTML Tabelle (nur Überschrift) des Alarmpuffers
Alarms.TableHeadUser	HTML Tabelle (nur Überschrift) der AlarmS
Alarms.TableBodyBuffer	HTML Tabelle (nur Inhalt) des Alarmpuffers
Alarms.TableBodyUser	HTML Tabelle (nur Inhalt) der AlarmS
Alarms.TableBuffer	HTML Tabelle des Alarmpuffers
Alarms.UserAlarmCnt	Anzahl AlarmS

3.8.3.10 Gruppe AlarmsDrv

Informationen zur Antriebsalarmtabelle

Die Gruppe AlarmsDrv liefert Informationen über die anstehenden Antriebsalarme. Auf alle Variablen kann nur lesend zugegriffen werden.

Tabelle 3- 32 Variablen der Gruppe AlarmsDrv

Variable	Beschreibung
AlarmsDrv.AlarmCnt	Anzahl der Antriebsalarme
AlarmsDrv.Table	HTML Tabelle mit allen anstehenden Antriebslarmen
AlarmsDrv.TableHead	Tabellenkopf für die HTML-Tabelle der anstehenden Antriebsalarme
AlarmsDrv.TableHeadBuffer	HTML Tabelle (nur Überschrift) des Antriebsalarmpuffers
AlarmsDrv.TableHeadUser	HTML Tabelle (nur Überschrift) der AlarmS des Antriebsalarmpuffers
AlarmsDrv.TableBodyBuffer	HTML Tabelle (nur Inhalt) des Antriebsalarmpuffers
AlarmsDrv.TableBodyUser	HTML Tabelle (nur Inhalt) der AlarmS der Antriebsalarme
AlarmsDrv.TableBuffer	HTML Tabelle des Antriebsalarmpuffers
AlarmsDrv.UserAlarmCnt	Anzahl AlarmS der Antriebsalarme

3.8.3.11 Gruppe ActiveTraces

Variablen der Gruppe ActiveTraces

Die Gruppe ActiveTraces liefert sowohl die Anzahl der aktiven Traces als auch eine Liste der aktiven Traces. Auf alle Variablen kann nur lesend zugegriffen werden.

Tabelle 3- 33 Variablen der Gruppe ActiveTraces

Variable	Beschreibung
ActiveTraces.TraceCnt	Anzahl der aktiven Traces
ActiveTraces.TableHead	Liefert den Kopf einer HTML-Tabelle mit allen aktiven Traces. Der Inhalt lautet: <tr><th>Name</th><th>State</th></tr>
ActiveTraces.Table	Liefert den Inhalt der HTML-Tabelle mit allen aktiven Traces. Jede Zeile ist nach dem folgenden Format aufgebaut: <tr><td>NAME</td><td>STATE</td></tr> **Hinweis:** Die im Format angegebenen Platzhalter NAME und STATE werden durch den entsprechenden Wert des jeweiligen Traces ersetzt.

3.8.3.12 Gruppe Watch

Variablen der Gruppe Watch

Die Gruppe Watch liefert den Zugriff auf gespeicherte Watch-Tabellen. Auf alle Variablen kann nur lesend zugegriffen werden.

Tabelle 3- 34 Variablen der Gruppe Watch

Variable	Beschreibung
Watch.TableNames	Kommaseparierte Liste der Namen der Watch-Tabellen
Watch.TableHead	Tabellenkopf für die HTML-Tabelle einer Watch-Tabelle
Watch.TablesCount	Anzahl der Watch-Tabellen
Watch.Tables.*TableName*.csv	Export der angegebenen Watch-Tabelle (*TableName*) als CSV-Datei
Watch.Tables.*TableName*.xml	Export als XML-Datei der angegebenen Watch-Tabelle (*TableName*) zur Übertragung auf andere Steuerungen
Watch.Tables.*TableName*.html	Liefert die angegebene Watch-Tabelle (*TableName*) im HTML-Format

3.8.3.13 Vergleich zur Gerätediagnose des SIMOTION SCOUT

Vergleich Gerätediagnose in SIMOTION SCOUT

Die in diesem Kapitel beschriebenen Variablen sind an die Sicht der Gerätediagnose im SIMOTION SCOUT angelehnt. Folgende Bilder zeigen den Zusammenhang zwischen den "SIMOTION diagnostics" Variablen und der Gerätediagnose im SIMOTION SCOUT.

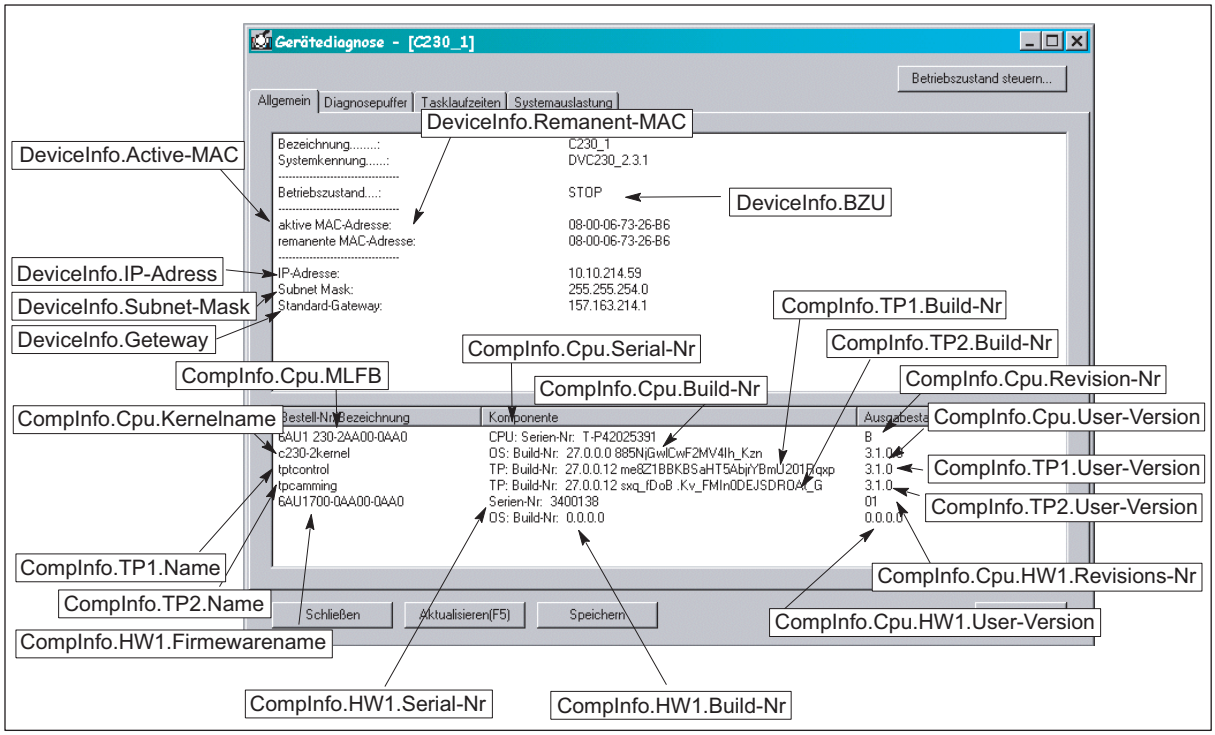


Bild 3-90 Gerätediagnose "Allgemein"

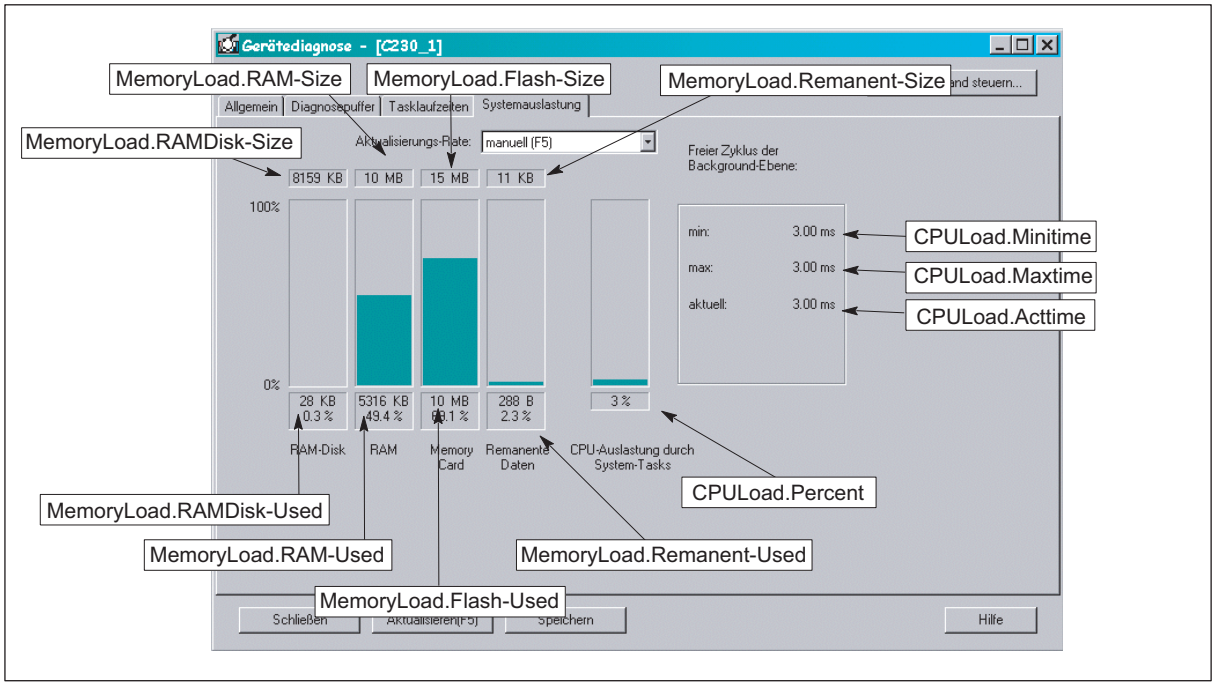


Bild 3-91 Gerätediagnose "Systemauslastung"

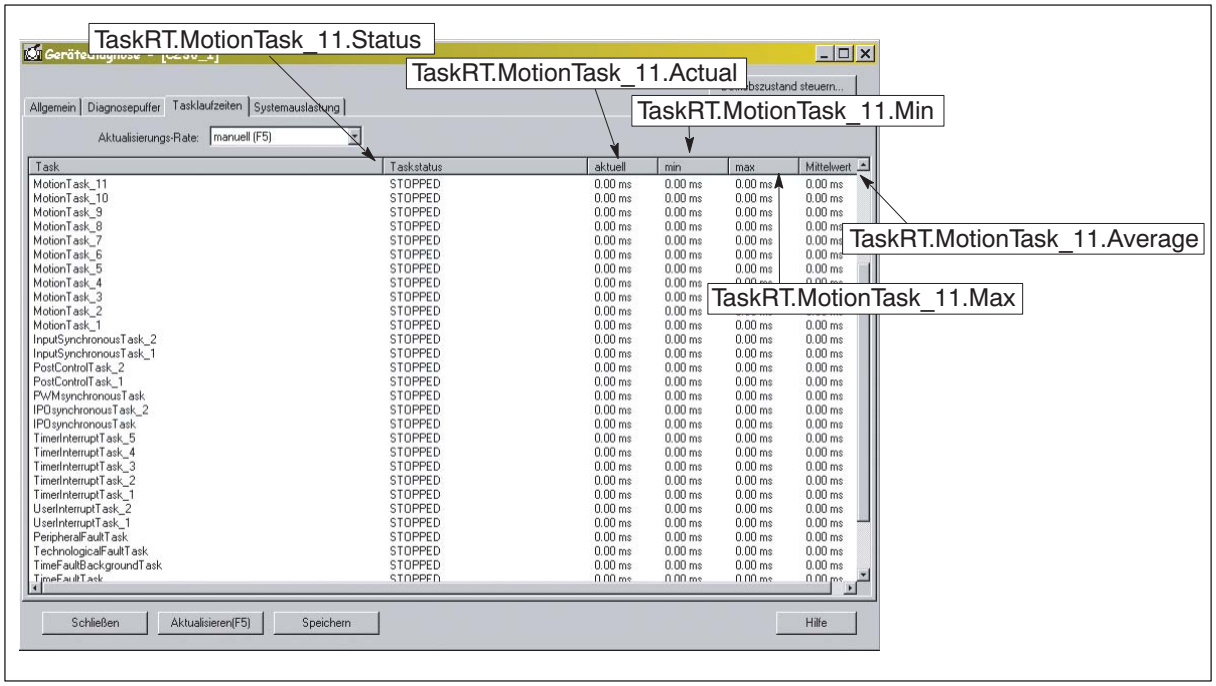


Bild 3-92 Gerätediagnose "Tasklaufzeiten"

3.8.4 UserConfig

3.8.4.1 Benutzerdefinierte Variablen

Die benutzerdefinierten Variablen werden in der WebCfg.xml (Seite 276) vereinbart und können im Variablen Provider gelesen werden.

Für den **Variablen Provider** lautet die **ItemName**-Syntax:

ItemName=" constants/VariablenName"

Einige konstante Variablen sind in IT DIAG vorinstalliert:

Name	Typ	Erklärung
ForceUserMsgLanguageID	Ganzzahl (LCID)	Legt die Sprache fest, die beim Import von benutzerdefinierten Meldungen (Diagnosepuffer bzw. AlarmS) verwendet wird. Spracheinstellung der AlarmS- und der benutzerdefinierten Diagnosepuffer-Meldungen (Seite 22)
WatchWritable	YES / NO Voreinstellung: YES	Legt fest, ob Watch-Tabellen auf den Standard-Seiten, editiert und gelöscht werden dürfen.
BasicWatchWritable	YES / NO Voreinstellung: YES	Legt fest, ob Watch-Tabellen auf den Basic-Seiten, editiert und gelöscht werden dürfen.
UserArea	Zeichenkette: Embedded, EmbeddedSimple, StandAlone	Anzeigeart der User's Area. Eingebettete anwenderdefinierte Seiten (Seite 126)
UserDir	Zeichenkette	Verzeichnis für User-Seiten: "/FILES" + <UserDir>

Übersicht vorinstallierter konstanter Variablen

Siehe auch

IT DIAG Configuration data (Seite 80)

3.9 Secure Socket Layer

Einleitung

Durch das Secure Socket Layer Protokol (SSL) wird eine verschlüsselte Datenübertragung zwischen einem Client und der SIMOTION ermöglicht. Das Secure Socket Layer Protokoll bildet die Basis für HTTPS-Zugriffe des Browsers auf die SIMOTION Steuerung.

Der verschlüsselte Zugriff auf eine SIMOTION kann sowohl über SIMOTION IT OPC XML-DA als auch über SIMOTION IT DIAG erfolgen.

In diesem Kapitel erfahren Sie, welche Schritte Sie durchführen müssen, um eine verschlüsselte Datenübertragung zwischen einem Client und der SIMOTION zu ermöglichen. Dabei bestehen die folgenden Möglichkeiten:

1. Sie verfügen über eine Certification Authority (CA) in Ihrem Unternehmen und Ihnen liegen die benötigten Schlüsseldateien vor. In diesem Fall ist für Sie das Kapitel "Schlüsseldateien auf die SIMOTION übertragen" relevant.
2. Sie verfügen über keine CA in Ihrem Unternehmen. In diesem Fall müssen Sie die Schlüsseldateien selbst erstellen. Die Vorgehensweise ist exemplarisch in Erstellen von Schlüsseldateien (Seite 267) beschrieben. Nach der Erstellung müssen die Schlüsseldateien auf die SIMOTION übertragen werden.

Hinweis

HTTPS Verbindungen werden ab SIMOTION V3.2 unterstützt.

3.9.1 Schlüsseldateien

Verschlüsselungsverfahren

Für das dem Secure Socket Layer Protokoll zu Grunde liegende Verschlüsselungsverfahren benötigen Sie zwei Schlüsseldateien. Zum einen benötigen Sie ein öffentliches Zertifikat (Public Certificate), zum anderen einen privaten Schlüssel (Private Key). Das Schlüsselpaar wird individuell für die entsprechende SIMOTION Steuerung erstellt. Dadurch wird beim HTTPS-Zugriff sichergestellt, dass die angeforderte Adresse auch wirklich der erreichten SIMOTION Steuerung entspricht.

Hinweis

Der verschlüsselte Zugriff auf die SIMOTION Steuerung ist ausschließlich über den bei der Schlüsselerstellung angegebenen Bezeichner (Name / IP-Adresse) der Steuerung möglich.

Weitere Informationen zu Secure Socket Layer Zertifikaten erhalten Sie unter http://www.verisign.de.

Auslieferungszustand

Damit Sie im Auslieferungszustand der SIMOTION IT DIAG (Diagnose-Standardseiten) per HTTPS auf die SIMOTION Steuerung zugreifen können, werden zwei in die WebCfg.xml integrierte Schlüsseldateien mit ausgeliefert.

Bei einem HTTPS-Zugriff unter Verwendung der mitgelieferten Schlüsseldateien erhalten Sie eine Warnung, da das Zertifikat unbekannt ist und die aktuell verwendete Adresse der Steuerung nicht dem Namen der Steuerung im Zertifikat entspricht.

3.9.2 Schlüsseldateien auf die SIMOTION Steuerung übertragen

Um den IT DIAG Zugriff über HTTPS auf das SIMOTION Gerät zu ermöglichen, müssen Sie die Schlüsseldateien "MWSSLCer.pem" und "MWSSLKey.pem" auf die Speicherkarte der SIMOTION Steuerung übertragen.

Für das Kopieren der Schlüsseldateien auf die Speicherkarte benötigen Sie ein Schreib-/Lesegerät für Speicherkarten.

Die Vorgehensweise zum Übertragen der Schlüsseldateien auf die SIMOTION P350 Steuerung wird separat beschrieben.

Vorgehensweise

1. Legen Sie die Speicherkarte Ihrer SIMOTION Steuerung in das Schreib-/Lesegeräte für Speicherkarten ein.
2. Kopieren Sie die Dateien "MWSSLCer.pem" und "MWSSLKey.pem" auf der Speicherkarte in das Verzeichnis \USER\SIMOTION\HMICFG. Falls das Verzeichnis nicht existiert, legen Sie dieses an.
3. Legen Sie die Speicherkate in die SIMOTION Steuerung ein und schalten Sie diese ein.

Nach dem Hochlauf der SIMOTION Steuerung steht Ihnen der OPC XML-DA Zugriff über HTTPS fehlerfrei zur Verfügung.

Vorgehensweise bei P350

1. Beenden Sie die SIMOTION P Steuerung.
2. Kopieren Sie die Dateien "MWSSLCer.pem" und "MWSSLKey.pem" auf Ihrer P350 in das Verzeichnis F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG (Pfadangabe bei Default-Installation).
3. Starten Sie die SIMOTION P Steuerung.

Nach dem Hochlauf der SIMOTION Steuerung steht Ihnen der OPC XML-DA Zugriff über HTTPS fehlerfrei zur Verfügung.

3.9.3 Erstellen von Schlüsseldateien mit Script (ab V4.1)

Überblick

Hinweis

HTTPS-Verbindungen werden ab SIMOTION V3.2 unterstützt.

Sollte keine Certification Authority (CA) in Ihrem Unternehmen vorhanden sein, so empfehlen wir Ihnen die in diesem Kapitel beschriebene Vorgehensweise. Die Erstellung des Zertifikates und der Schlüsseldateien erfolgt mittels des Tools OpenSSL und einem Perl Script.

Folgende Schritte sind durchzuführen:

Nr.	Arbeitsschritt	Anmerkung
1.	Installation der Perl Laufzeitumgebung	Falls kein Perl vorhanden
2.	Installation von OpenSSL	
3.	Zertifikat und Schlüsseldateien mit dem Perl Script erstellen	
4.	WebCfg.xml in diesen Ordner kopieren	Eine Standard WebCfg.xml befindet sich z. B. beim SIMOTION SCOUT V4.1 auf der DVD "SIMOTION SCOUT Add-on" im Ordner AddOn\4_Accessories\Simotion_IT\2_Configuration.
5.	Perl Script ausführen	Das SIMOTION Perl Script mit entsprechenden Optionen aufrufen.
6.	Erstellte Datei WebCfg.xml mit einem Browser auf die Steuerung laden	Dieser Schritt muss je Steuerung einmal ausgeführt werden.
7.	Erstelltes Zertifikat in den Browser des PCs importieren	Dieser Schritt muss je PC einmal ausgeführt werden.

Nach dem Hochlauf der SIMOTION Steuerung steht Ihnen der HTTPS-Zugriff zur Verfügung.

Installation der Perl Laufzeitumgebung

Falls keine Perl Laufzeitumgebung auf Ihrem PC vorhanden ist installieren Sie Perl. Ein kostenloses Setup für Windows finden Sie z. B. auf folgenden Internetseiten:

- http://www.activestate.com
- http://www.perl.org

Installation von OpenSSL

Ein kostenloses Setup für Windows finden Sie z. B. auf folgenden Internetseiten:

- http://openssl.org

Hinweis

Es wird davon ausgegangen, dass OpenSSL nach C:\OpenSSL installiert wird. Sollte ein anderer Pfad gewählt worden sein, dann muss die Zeile 5 des Perl-Scripts entsprechend angepasst werden.

Zertifikat und Schlüsseldateien mit dem Perl Script erstellen

- Legen Sie einen beliebigen Ordner auf Ihrem lokalen Laufwerk an, z. B. "c:\SimotionSSL".
- Kopieren Sie die Perl Scriptdatei "cert.pl" in den erstellten Ordner.
  Die Perl Scriptdatei finden Sie auf der DVD "SIMOTION SCOUT Add-on" im Ordner AddOn\4_Accessories\Simotion_IT\6_Tools.
- Kopieren Sie eine Standard "WebCfg.xml" Datei in den erstellten Ordner.
  Eine Standard Vorlage der Datei "WebCfg.xml" finden Sie auf der DVD "Utilities & Applications" oder wird auch automatisch nach dem Hochlauf der Steuerung auf dem Speicherkärtchen angelegt.
- Führen Sie das Perl Script mit folgenden Optionen aus:
  "perl cert.pl -c <IP-Adresse> -p "

  Es wird eine CA angelegt und anschließend ein Server-Schlüssel und Zertifikat erzeugt und das Zertifikat signiert.

  Folgende Dateien werden in dem Ordner (z. B. "c:\SimotionSSL") abgelegt:

  "c:\SimotionSSL\CA\cakey.pem"
  "c:\SimotionSSL\CA\cacert.pem"
  "c:\SimotionSSL\out\<IP-Address>\MWSSLKey.pem"
  "c:\SimotionSSL\out\<IP-Address>\MWSSLCert.pem"
  "c:\SimotionSSL\out\<IP-Address>\WebCfg.xml"

  Hinweis

  Hilfe zum Aufruf erhalten Sie mit der Option -h: "perl cert.pl -h"

- Erstellte Datei "WebCfg.xml" mit einem Browser auf die Steuerung laden

  Die Datei WebCfg.xml enthält das Schlüsselpaar (Server-Schlüssel und Zertifikat) für den HTML-Server der Steuerung.

- Importieren des Zertifikates in den Browser

  Das Zertifikat cacert.pem kann dem PC durch Importieren im Browser bekannt gemacht werden. Wird das Zertifikat nicht importiert, kommt beim Öffnen des Browsers eine Meldung, dass die signierte CA nicht bekannt ist.

  Siehe auch Importieren des Zertifikats in den Browser (Seite 269).

3.9.3.1 Importieren des Zertifikats in den Browser

Wenn Sie SSL mit einer eigenen Zertifizierungsstelle betreiben, müssen Sie Ihre PCs zur Kommunikation mit der SIMOTION Steuerung vorbereiten. Hierfür müssen Sie das Zertifikat "SIMOTION.cer" in die Liste der Root Zertifikate aufnehmen.

Vorgehensweise

Zunächst müssen Sie die Datei cacert.pem wie folgt bearbeiten:

1. Kopieren Sie die Datei "cacert.pem" (im Beispiel unter "Eigene Dateien\OpenSSL\demoCA").
2. Fügen Sie die kopierte Datei unter z. B. "Eigene Dateien\OpenSSL\" ein.
3. Benennen Sie die Datei um in "SIMOTION.cer".

Das Importieren des Zertifikates entnehmen Sie bitte der Anleitung ihres Browsers.

Liste der Abkürzungen

4

4.1 Liste der Abkürzungen

Abkürzungen

CA	Certification Authority
CSS	Cascading Style Sheets
CSV	Character Separated Values
DO	Drive Object (Antriebsobjekt)
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
FTP	File Transfer Protocol
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
JS	Javascript
MWSL	MiniWeb Server Language
OPC	Bezeichnet eine Standardschnittstelle für die Kommunikation in der Automatisierungstechnik. http://www.opcfoundation.org/
OPC XML-DA	OPC XML Data Access
SSL	Secure Socket Layer
TO	Technology Object (Technologieobjekt)
TVS	Trace Via SOAP
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformation

Anhang

5

5.1 WebCfg.xml

5.1.1 <ALTERNATE_PORTNUMBER>

Tag	<ALTERNATE_PORTNUMBER>
	Zusätzlicher Port für Anfragen an den Webserver. Jeder TCP/IP Server (bzw. Service) verfügt über eine so genannte Well-Known Portnummer, unter der er von einem Client ansprechbar ist. Für Webserver ist dies im Regelfall die Portnummer 80. Der Webserver kann zusätzlich auf einer zweiten Portnummer "horchen". Kommt ein Request auf dieser Portnummer, so wird im ECB das Flag WEB_FLAG_ALTERNATIVE_PORT gesetzt. In Verbindung mit einer Firewall kann hiermit zum Beispiel ein Sicherheitskonzept etabliert werden, das durch die Firewall gesteuert wird. Eine andere Anwendung dieses alternativen Ports verwendet das DAV Modul, das damit erkennt, ob es sich um einen DAV-Request oder um einen Web-Request handelt.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<ALTERNATE_PORTNUMBER>`
	`81`
	`</ALTERNATE_PORTNUMBER>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In diesem Beispiel wird die alternative Portnummer des Webservers auf 81 gesetzt.

Siehe auch

Überblick (Seite 106)

5.1.2 <ALTERNATE_SSL_PORTNUMBER>

Tag	<ALTERNATE_SSL_PORTNUMBER>
	Für das SSL-Protokoll (Secure Socket Layer) wird eine weitere Well-Known Portnummer gebraucht. Dies ist im Regelfall die Portnummer 443. Der Webserver kann zusätzlich auf einer zweiten Portnummer "horchen". Kommt ein Request auf dieser Portnummer, so wird im ECB das Flag MWEB_FLAG_ALTERNATIVE_PORT gesetzt. In Verbindung mit einer Firewall kann hiermit zum Beispiel ein Sicherheitskonzept etabliert werden, das durch die Firewall gesteuert wird. Eine andere Anwendung dieses alternativen Ports verwendet das DAV Modul, das damit erkennt, ob es sich um einen DAV-Request oder um einen Web-Request handelt. Dieses ist der Alternative SSL-Port.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<ALTERNATE_SSL_PORTNUMBER>`
	`5443`
	`</ALTERNATE_SSL_PORTNUMBER>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In diesem Beispiel wird die alternative Portnummer für SSL auf 5443 gesetzt.

5.1.3 <BASE>

Tag	<BASE>
	Im <BASE>-Tag der WebCfg.xml werden die Linklisten der anwenderdefinierten HTML-Seiten hinterlegt.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE LOCALLINK="/">`
	`[...]`
	`<index.mbs LOCALLINK="mydir/index.mbs" PREFER_EXTERNAL="TRUE"/>`
	`[...]`
	`</BASE>`
	`[...]`
	`</SERVERPAGES>`

5.1.4 <BROWSEABLE>

Tag	<BROWSEABLE>
Werte	TRUE, FALSE
	Durchsuchen und Anzeigen von Verzeichnissen an- bzw. abschalten. Mit diesem Tag kann das Browsen global für alle Verzeichnisse erlaubt werden. In diesem Fall werden die einzelnen `BROWSEABLE` Attribute bei den `LOCALLINKS` unbedeutend.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<BROWSEABLE>`
	`FALSE`
	`</BROWSEABLE>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In diesem Beispiel ist das globale Browsen abgeschaltet und kann explizit für einzelne Links eingeschaltet werden. Dies ist das Default-Verhalten.

Siehe auch

Browsen von Verzeichnissen (Seite 111)

5.1.5 <CONFIGURATION_DATA>

Tag	<CONFIGURATION_DATA>
	Jedes Modul hat die Möglichkeit modulspezifische Konfigurationsdaten innerhalb dieses Tags zu definieren. Diese Daten können über den Default Service wieder ausgelesen werden. Das Format der einzelnen Konfigurationsdaten hängt ausschließlich von den Modulen ab. Es kann daher nicht allgemein beschrieben werden.

Beispiel	`<SERVERPAGES>`
	`  [...]`
	`  <CONFIGURATION_DATA>`
	`  <USERCONFIG>`
	`  <UserArea>EmbeddedSimple</UserArea>`
	`  <UserDir/>`
	`  <IncludeScriptsDirectly>NO</IncludeScriptsDirectly>`
	
	`  <!-- Add your constants here -->`
	`  <ForceUserMsgLanguageID>1031</ForceUserMsgLanguageID>`
	`  </USERCONFIG>`
	`  </CONFIGURATION_DATA>`
	`  [...]`
	`</SERVERPAGES>`

5.1.6 <DEFAULTDOCUMENT>

Tag	<DEFAULTDOCUMENT>
	Angabe des Dokuments, das angezeigt werden soll, wenn die vom Browser empfangene URL keine explizite Seitenangabe enthält. Diese heißen oft Default.mcs, oder Index.mcs. Es kann nur ein Default-Dokument geben. Wird kein Default-Dokument gefunden, und ist File Browsing gestattet, so wird das Verzeichnis selbst zurückgegeben.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<DEFAULTDOCUMENT>`
	`Default.mcs`
	`</DEFAULTDOCUMENT>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	Wird z. B. mit der URL http://Servername/MyDir ein Verzeichnis abgefragt, hängt der Webserver den Datei Namen "Default.mcs" an die URL an (http://Servername/MyDir/Default.mcs) und versucht diese dann aufzulösen:
	• Gelingt dies, so wird Default.mcs an den Client zurückgesendet.
	• Gelingt dies nicht, wird entweder eine Verzeichnisansicht zurückgesendet, oder eine HTTP 404 "Not Found" Fehlermeldung (Je nach Konfiguration).

5.1.7 <MIME_TYPES>

Tag	<MIME_TYPES>
	Der Webserver bietet mit der Mime-Type Tabelle eine Möglichkeit, die Dateiendung einer Datei auf einen dazugehörigen Mime-Type zu mappen. Zusätzlich kann für den Verzeichnis Browser ein Icon für jede Dateiendung hinterlegt werden, sodass jede Dateiendung ein anderes Icon bekommt.
Erläuterung	Im Dateisystem wird der Inhalt einer Datei durch seine Dateiendung (z. B. "txt" für Text Dateien) gekennzeichnet. Eine solche ist in einem Transport Protokoll wie HTTP nicht zwingend möglich. Daher wurde ein HTTP-Header namens "Mime-Type" eingefügt, der genau so eine Information über den Inhaltstyp enthält.
Beispiel	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE> [...] </BASE> <SERVEROPTIONS> <MIME_TYPES> <FILE EXTENSION="htm" MIMETYPE="text/html" ICON="/Images/www.gif" FILTER="TRUE"/> <FILE EXTENSION="html" MIMETYPE="text/html" ICON="/Images/www.gif" FILTER="FALSE"/> [...] </MIME_TYPES> [...] </SERVEROPTIONS> [...] </SERVERPAGES> Für die Dateiendungen "htm" und "html" wird der Mime-Type "text/html" festgelegt. Im Verzeichnis Browser wird das Icon mit der URL "/Images/www.gif" für die Kennzeichnung dieser Dateitypen verwendet. Mit dem FILTER-Attribut kann angegeben werden, ob Dateien mit dieser Dateiendung gefiltert werden, oder nicht. Weitere Informationen zu Mime-Typen finden sich in den RFCs 2045 ff.

5.1.8 <PORTNUMBER>

Tag	<PORTNUMBER>
	Jeder TCP/IP Server (bzw. Service) verfügt über eine so genannte Well-Known Portnummer, unter der er von einem Client ansprechbar ist. Für Webserver ist dies im Regelfall die Portnummer 80. Unter dem Tag <PORTNUMBER> kann diese Portnummer eingestellt werden. Wird nichts eingestellt, so wird automatisch die Nummer 5001 eingestellt, um nicht mit einem evtl. schon vorhandenen Webserver zu kollidieren.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`  [...]`
	`  <BASE>`
	`  [...]`
	`  </BASE>`
	`  <SERVEROPTIONS>`
	`  <PORTNUMBER>`
	`  80`
	`  </PORTNUMBER>`
	`  [...]`
	`  </SERVEROPTIONS>`
	`  [...]`
	`</SERVERPAGES>`
	
	In diesem Beispiel wird die Portnummer des Webservers auf 80 gesetzt.

5.1.9 <SERVEROPTIONS>

Tag	<SERVEROPTIONS>
	Das Tag "Server Optionen" umschließt alle grundlegenden Parameter des Webserver. Die innerhalb des Tags gemachten Einstellungen betreffen den Kern des Webservers.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`  [...]`
	`  <BASE>`
	`  [...]`
	`  </BASE>`
	`  <SERVEROPTIONS>`
	`  [...]`
	`  </SERVEROPTIONS>`
	`  [...]`
	`</SERVERPAGES>`

5.1.10 <SSLPORTNUMBER>

Tag	<SSLPORTNUMBER>
	Für das SSL-Protokoll (Secure Socket Layer) wird eine weitere Well-Known Portnummer gebraucht. Dies ist im Regelfall die Portnummer 443. Wird SSL im Webserver verwendet, kann hier die Portnummer für SSL eingestellt werden. Wird nichts eingestellt, so wird automatisch die Nummer 5443 eingestellt, um nicht mit einem evtl. schon vorhandenen Webserver zu kollidieren.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<SSLPORTNUMBER>`
	`443`
	`</SSLPORTNUMBER>`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	
	In diesem Beispiel wird die Portnummer für SSL auf 443 gesetzt.

5.1.11 <TIMEZONE>

Tag	<TIMEZONE>
	Einstellung der Zeitzone des Webservers. Um eine Zeitzonensynchronisation zu anderen Partnern machen zu können, sprich um die lokal eingestellte Uhrzeit des Webservers auf UTC umrechnen zu können, muss der Webserver wissen, auf welche Zeitzone die lokale Uhr der Steuerung eingestellt ist. Als Wert wird hier die Abweichung in +/- Minuten zu UTC angegeben.

Beispiel	`<?xml version="1.0" standalone="yes"?>`
	`<SERVERPAGES>`
	`[...]`
	`<BASE>`
	`[...]`
	`</BASE>`
	`<SERVEROPTIONS>`
	`<TIMEZONE>`
	`+60`
	`</TIMEZONE >`
	`[...]`
	`</SERVEROPTIONS>`
	`[...]`
	`</SERVERPAGES>`
	In diesem Beispiel wird Zeitzone auf "UTC + 60 Minuten" gesetzt. Das entspricht MEZ Winterzeit.

5.1.12 <USERDATABASE>

Tag	<USERDATABASE>
	Verschiedenste Bereiche des Webservers, angefangen von HTML-Seiten, Verzeichnissen etc. bis hin von einzelnen Aktionen von Applikationen können mit einem Zugriffsschutz geschützt werden. Das Sicherheitssystem ist folgendermaßen aufgebaut: • es gibt Benutzer (User) • jeder User hat ein Passwort • es gibt Sicherheitsbereiche (SecureGroups bzw. Realms) • jeder Sicherheitsbereich hat eine Gruppe von Benutzern, die diesen "Betreten" dürfen • ein Benutzer kann in unterschiedlichen Sicherheitsbereichen Zutritt haben

Beispiel	`<UserDataBase>`
	`<USER NAME="Gast" PASSWORD="MyPassword">`
	`<DESCRIPTION>Gast User</DESCRIPTION>`
	`<GROUP Name="User">`
	`</USER>`
	`<USER NAME="Administrator">`
	`<DESCRIPTION>Administrator</DESCRIPTION>`
	`<GROUP NAME="MiniWeb Administratoren"/>`
	`<GROUP NAME="Administrator"/>`
	`<GROUP NAME="User"/>`
	`<GROUP NAME="FileAdministrator"/>`
	`<GROUP NAME="NoAccess"/>`
	`</USER>`
	`</UserDataBase>`
	
	Gegeben sei die folgende Link Struktur:
	`<BASE>`
	`<Trap1 LINK="/Trap2" SECUREGROUP="Member_Trap1"/>`
	`<Trap2 SECUREGROUP="Member_Trap2"/>`
	`<Winner.mcs>`
	`[ . . . ]`
	`</Winner.mcs>`
	`</Trap2></BASE>`
	Ein Benutzer, der Mitglied der beiden SecureGroups "Member_Trap1" und "Member_Trap2" ist, kann die URL
	http://Server/Trap1/Winner.mcs
	nicht anfordern, da Trap1 die SecureGroup "Member_Trap1" erfordert und wenn diese vorhanden ist, auf Trap2 verweist (das ist eine Security Verletzung). Diese erfordert jetzt aber "Member_Trap2". Dieser zweiten Gruppe gehört der Benutzer zwar an, aber er hat sich bereits in der Rolle "Member_Trap1" angemeldet. Daher wird dieser zweite Request abgelehnt.
	Der direkte Zugriff auf
	http://Server/Trap2/Winner.mcs
	ist dagegen möglich, da der Benutzer in der SecureGroup "Member_Trap2" ist, und nur diese angefordert wird.

5.1.13 Attribut BROWSEABLE

Tag	LOCALLINK oder als globaler Schalter über das Tag <BROWSEABLE>	
Attribut	BROWSEABLE	BROWSEABLE kann "TRUE" oder "FALSE" zugewiesen werden. Wenn ein Client diesen Link anspricht, wird eine Verzeichnisansicht des Verzeichnisses erzeugt. Ab diesem Verzeichnis kann auch in Unterverzeichnisse navigiert werden. Sie gelangen auch in weiter oben gelegene Verzeichnisse, wenn für diese das Browsen auch erlaubt ist. Sie können, vorausgesetzt die entsprechenden Rechte sind vorhanden, Dateien senden, empfangen und löschen, sowie Verzeichnisse anlegen und löschen. Das Aussehen des Verzeichnisses im Client ist frei konfigurierbar. Bei der Default Applikation <DEFAPP> wird dies genauer beschrieben. Wenn kein Authentifizierungsmechanismus im Webserver vorhanden, sind verändernde Zugriffe generell nicht gestattet.
Beispiel	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE> <www LOCALLINK="/UserData" BROWSEABLE="TRUE" REALM="Bediener"/> <Test.mcs LINK="/Tests/Test.mcs/"/> <Default.mcs> <![CDATA[<HTML> <HEAD> [...]]]> </Default.mcs> </BASE> [...] </SERVERPAGES>	

5.1.14 Attribut LOCALLINK

Tag	Beliebiger Knoten: <BASE> etc.	
Attribut	LOCALLINK	Lokale Links sind die einzige Möglichkeit auf das physikalische Datei System zuzugreifen. Jeder Datenknoten des XML-Dateisystems kann ein LOCALLINK Attribut besitzen, auch der <BASE> Knoten. Der <BASE> Knoten entspricht dem Root Eintrag des Dateisystems. Wurde ein LOCALLINK Attribut gefunden, so wird trotzdem weiterhin versucht, das XML-Dateisystem zu parsen. Das XML-Dateisystem hat Vorrang vor dem externen Dateisystem. Dies gilt für lesende und schreibende Zugriffe. Diese Priorität kann mit dem Attribut PREFER_EXTERNAL geändert werden, sodass das externe Dateisystem Vorrang vor dem XML-Dateisystem hat. Hinweis: Unter Umständen können bereits im Dateisystem vorhandene Dateien durch diese Vorrangigkeit überdeckt werden.

5.1.15 Attribut MODIFY

Tag	Beliebiger Knoten: BASE, MainDir, etc.	
Attribut	MODIFY	Besitzt ein Verzeichnis ein MODIFY Attribut, und ist der eingeloggte User Mitglied einer der angegebenen Gruppen, so darf der User in diesem Verzeichnis alle Schreiboperationen ausführen. Er darf • neue Verzeichnisse anlegen • Dateien überschreiben • Dateien löschen • neue Dateien anlegen Der User muss auf dem Verzeichnis natürlich auch READ Rechte haben, sonst hätte er erst keinen Zugriff auf das Verzeichnis.

5.1.16 Attribut PREFER_EXTERNAL

Tag	LOCALLINK	
Attribut	PREFER_EXTERNAL	Das Attribut PREFER_EXTERNAL kann nur in Verbindung mit LOCALLINK gesetzt werden. Wenn PREFER_EXTERNAL gesetzt ist, werden Zugriffe (Lesen / Schreiben) vorrangig auf dem externen Dateisystem durchgeführt. Existiert eine Datei mit demselben Zugriffspfad gleichzeitig im XML und externem Dateisystem, dann wird standardmäßig die Datei aus dem XML-Dateisystem genommen. Ist PREFER_EXTERNAL gesetzt, wird die Datei aus dem externen Dateisystem genommen. Dasselbe gilt für Verzeichnisse (im Schreibfall).

5.1.17 Attribut READ

Tag	Beliebiger Knoten: `BASE, MainDir`, etc.	
Attribut	`READ`	Ist an einem Verzeichnis ein `READ` Attribut angegeben, muss der User Mitglied einer der bei `READ` angegebenen Gruppen sein. Bei `READ` können mehrere Gruppen angegeben werden, diese müssen mit Komma getrennt sein, es dürfen keine Whitespace Zeichen verwendet werden.
Beispiel	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE LOCALLINK="/"> <MainDir REALM="USER" LOCALLINK="/Base/" > <www LOCALLINK="/WebSeiten/" BROWSEABLE="TRUE" READ="Administrator" WRITE="FileAdministrator" /> </MainDir> <Test.mcs LOCALLINK="/Tests/Test.mcs/"/> <XMLDir> </XMLDir> <Default.mcs> <![CDATA[<HTML> <HEAD> [...]]]> </Default.mcs> </BASE> [...] </SERVERPAGES>	

5.1.18 Attribut REALM

Tag	Beliebiger Knoten: `BASE`, `MainDir`, etc.	
Attribut	`REALM`	Mit `REALM`-Attribut wird ein Sicherheitsbereich eingerichtet. `REALM` darf nur einen Gruppennamen enthalten. Durch das `REALM`-Attribut wird ein Login für alle Benutzer einer Gruppe ermöglicht. Für alle Benutzer, die nicht dieser Gruppe angehören, ist der Zugriff gesperrt.
Beispiel	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE LOCALLINK="/"> <MainDir REALM="USER" LOCALLINK="/Base/" > <www LOCALLINK="/WebSeiten/" BROWSEABLE="TRUE" READ="Administrator" WRITE="FileAdministrator" /> </MainDir> <Test.mcs LOCALLINK="/Tests/Test.mcs/"/> <XMLDir> </XMLDir> <Default.mcs> <![CDATA[<HTML> <HEAD> [...]]]> </Default.mcs> </BASE> [...] </SERVERPAGES>	

5.1.19 Attribut WRITE

<table>
<tr><td>Tag</td><td colspan="2">Beliebiger Knoten: <code>BASE, MainDir,</code> etc.</td></tr>
<tr><td>Attribut</td><td><code>WRITE</code></td><td>Besitzt ein Verzeichnis ein <code>WRITE</code>-Attribut, und ist der eingeloggte User Mitglied einer der angegebenen Gruppen, so darf der User in diesem Verzeichnis nur neue Dateien anlegen.
Er darf
• keine neuen Verzeichnisse anlegen
• keine Dateien überschreiben
• keine Dateien löschen
• neue Dateien anlegen

Der User muss auf dem Verzeichnis natürlich auch <code>READ</code> Rechte haben, sonst hätte er erst keinen Zugriff auf das Verzeichnis.</td></tr>
<tr><td>Beispiel</td><td colspan="2"><pre>
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE LOCALLINK="/">
    <MainDir REALM="USER" LOCALLINK="/Base/" >
     <www LOCALLINK="/WebSeiten/"
        BROWSEABLE="TRUE"
        READ="Administrator"
        WRITE="FileAdministrator" />
    </MainDir>
    <Test.mcs LOCALLINK="/Tests/Test.mcs/"/>
    <XMLDir>
    </XMLDir>
    <Default.mcs>
     <![CDATA[
      <HTML>
       <HEAD>
       [...]
     ]]>
    </Default.mcs>
  </BASE>
  [...]
</SERVERPAGES>
</pre></td></tr>
</table>

5.2 WebCfgFrame.xml

5.2.1 <BASE>

Tag	<BASE>
	Im <BASE>-Tag der WebCfgFrame.xml werden die Linklisten der Standardseiten (HTML bzw. XML) hinterlegt.
Beispiel	<?xml version="1.0" standalone="yes"?> <SERVERPAGES> [...] <BASE LOCALLINK="/"> [...] <alarms.mbs LOCALLINK="html/standard/alarms.mbs" PREFER_EXTERNAL="TRUE"/> [...] </BASE> [...] </SERVERPAGES>

5.2.2 <CONVERSION>

<CONVERSION>

Tag	<CONVERSION>
	Der Konvertierungsservice bietet verschiedene Möglichkeiten, Daten von einem Format in ein anderes zu wandeln. Die Konvertierung wirkt in beide Richtungen. So sind die Wandlung von UTF8 nach ASCII, De- und Encoden von URLs bzw. HTML Seiten sowie typische Datentypumwandlungen, wie von Double nach String, möglich. Für die Umwandlung von UTF 8 nach ASCII wird eine Codepage verwendet, um die ASCII Zeichen >128 in beliebige UTF 8 Zeichen wandeln zu können.
Beispiel	<SERVERPAGES> [...] <CONVERSION> <CODEPAGE Name="Standard"> <CHAR UCS="F6" ASCII="F5" /> </CODEPAGE> </CONVERSION> [...] </SERVERPAGES> <CODEPAGE> kann beliebig viele <CHAR> Einträge haben. Allerdings machen mehr als 128 Einträge keinen Sinn, da ASCII nur 128 Einträge im Bereich 128 – 255 zur Verfügung stellt. Für jedes Mapping von Buchstaben wird eine Zeile in die Codepage eingetragen: <CHAR UCS="F5" ASCII="A0" /> UCS ist das UTF 8 Zeichen, das aus bis zu 4 Bytes bestehen kann. ASCII ist der Eintrag in der ASCII Tabelle, der diesem Zeichen entsprechen soll. Als "Standard" Codepage wird eine DOS nach UTF-8 Codepage mitgeliefert. Wenn das Zielsystem ANSI verwendet, so ist keine Codepage notwendig (z. B. unter Windows).

5.2.3 <DEFAPP>

Tag	<DEFAPP>
	Die Default Applikation hat folgende Aufgaben: • Senden von Dateien – HTML-Seiten – Images – Etc. • Empfangen einer neuen WebCfg.xml Konfigurationsdatei – Parsen dieser Datei – Restart des Webservers • Generieren des Verzeichnis Browsers – Erstellen der Verzeichnisinhalte – Löschen von Dateien – Erstellen bzw. Löschen von Verzeichnissen – Laden von Dateien Für einige dieser Aufgaben muss die Default Applikation selbst HTML Seiten generieren und zum Client zurücksenden. Für z. B. den Verzeichnis Browser oder die Bestätigung, wenn eine neue Datei empfangen wurde. Diese Seiten sollen frei gestaltbar sein, damit sie sich in das gewünschte Aussehen des Zielsystems einbinden lassen. Hierzu werden mehrere HTML-Fragmente definiert, mit deren Hilfe die Default Applikation die Antwort Seiten zusammenstellt. Diese HTML-Fragmente liegen in dem Konfigurationsbereich und sollen jetzt näher erläutert werden. Wie bereits im `<BASE>` Teil (Siehe Kapitel: Virtuelles Datei System (Seite 108)) gesehen, müssen die HTML-Fragmente in `<![CDADA[ .. ]]>` Blöcke gekapselt werden, da sie unter Umständen keine gültige XML Syntax aufweisen.

Beispiel	`<SERVERPAGES>`
	`[...]`
	`<CONFIGURATION_DATA>`
	`<DEFAPP>`
	`<DIRHEAD>`
	`[...]`
	`</DIRHEAD>`
	`<DIRTAIL>`
	`[...]`
	`</DIRTAIL>`
	`<DIRLINE>`
	`[...]`
	`</DIRLINE>`
	`<LOADSUCCEED>`
	`[...]`
	`</LOADSUCCEED>`
	`<SENDCOMPLETE>`
	`[...]`
	`</SENDCOMPLETE>`
	`</DEFAPP>`
	`[...]`
	`</CONFIGURATION_DATA>`
	`[...]`
	`</SERVERPAGES>`

Siehe auch

Anwenderdefinierte Verzeichnisseite (Seite 202)

5.2.4 <HTTP_RESULT_CODES>

Tag	<HTTP_RESULT_CODES>
	Alle Fehlermeldungen, die der Webserver erzeugt, können mit diesem Tag angepasst werden.

Beispiel	`<SERVERPAGES>`
	`[...]`
	`<HTTP_RESULT_CODES>`
	`<RESULT NAME="200" CODE="OK" LOCALLINK="/ErrorCodes/200_OK.mcs">`
	`</RESULT>`
	`<RESULT NAME="404" CODE="NOT FOUND">`
	`<![CDATA[`
	`<HTML>`
	`[...]`
	`</HTML>`
	`]]>`
	`[...]`
	`</HTTP_RESULT_CODES>`
	`[...]`
	`</SERVERPAGES>`
	Die Gliederung erfolgt nach den Fehlernummern (hier die Fehlernummer 404). Unter dem Tag `<RESULT>` befinden sich alle Einträge für diese Fehlernummer.
	Unter dem <CODE> Tag befindet sich die von HTTP spezifizierte Kurzfehlerzeile. Diese ist genormt und muss immer dieselbe sein, unabhängig von der Sprache. Für 404 ist dieser Text immer "NOT FOUND".
	Welche Fehlermeldungen es gibt, welche Kurztexte sie haben, und welche Bedeutung ist im RFC 2068 beschrieben.
	Der `<![CData[ ..]]>` Block enthält eine HTML-Seite, die im Fehlerfall an den Client gesendet wird, und eine Beschreibung des Fehlers enthalten sollte. Es gibt keine Einschränkungen, wie diese Seite aufgebaut sein muss. Z. B. könnte es sein, das sie nur eine Referenz auf die Eingangsseite des Webservers enthält, dieses ist beim Fehler 404 beliebt, um dem Client nicht zu zeigen, dass er auf eine nicht existierende Seite zugegriffen hat.

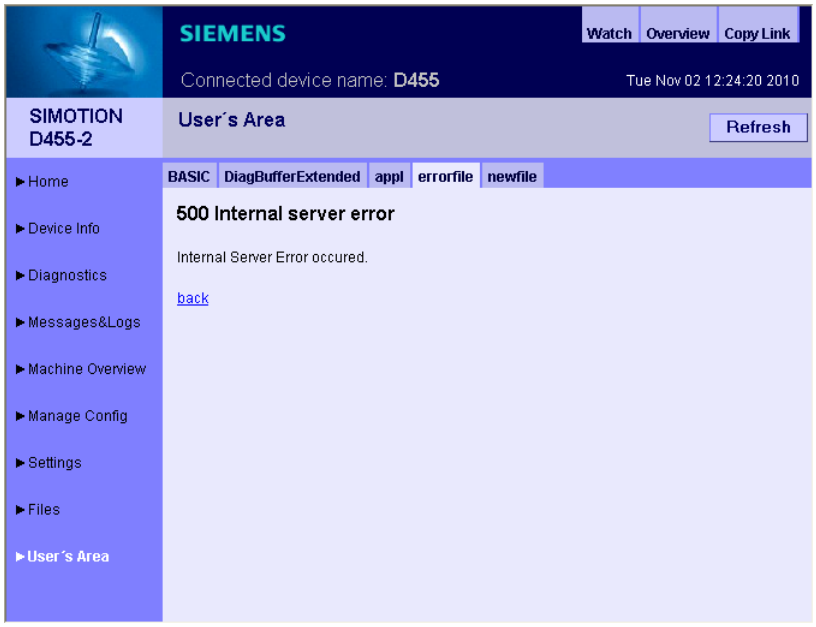


Bild 5-1 Ausgabe einer Fehlerseite

5.2.5 <SOAPAPP>

<SOAPAPP>

Tag	<SOAPAPP>
	<SOAPAPP> erlaubt die An- und Abschaltung von SOAP-Services. Durch das Löschen eines <WEBSERVICE> Tag kann der entsprechende Service dauerhaft abgeschaltet werden.
Beispiel	<SERVERPAGES> [...] <SOAPAPP> <STATIC> <WEBSERVICE NAME="OpcXml" URL="/SOAP/OPCXML" /> <WEBSERVICE NAME="TVS" URL="/SOAP/TVS"/> </STATIC> </SOAPAPP> [...] </SERVERPAGES>

5.2.6 <SSL>

Tag	<SSL>
	SSL bietet Kanalverschlüsselung für TCP/IP Anwendungen. Ist also nicht auf HTTP beschränkt. Aber SSL (oder der Nachfolger TLS) dürften gerade durch den Einsatz als Kanalverschlüsselung für HTTP bekannt geworden sein. Für die SSL-Verschlüsselung (wie für jede andere Verschlüsselung) werden Schlüssel benötigt. Bei SSL wird das Schlüsselmaterial vom Server bereitgestellt. Der <CERTIFICATES> Eintrag in der Konfiguration gibt an, wo dieses Schlüsselmaterial gefunden werden kann.
Beispiel	<SERVERPAGES> [...] <CONFIGURATION_DATA> <SSL> <CERTIFICATES DefaultCert="cert.pem" DefaultKey="key.pem" TrustedCAPath="CA" /> </SSL> [...] </CONFIGURATION_DATA> [...] </SERVERPAGES>

Das Schlüsselmaterial besteht aus zwei Teilen:

1. Dem Zertifikat, das auch weitere Informationen über den Server, wie z. B. den Servernamen enthält. Dieses enthält den öffentlichen Schlüssel des Servers. Nur das Zertifikat wird an den Client übertragen.
2. Dem privaten Schlüssel, den der Server verwendet, um Daten zu verschlüsseln bzw. Daten, die mit dem öffentlichen Schlüssel verschlüsselt wurden, zu entschlüsseln.

Die mit dem privaten Schlüssel verschlüsselten Daten können mit Hilfe des öffentlichen Schlüssels entschlüsselt werden.

Das Zertifikat ist in der Datei abgelegt, das unter dem Attribut DefaultCert abgelegt ist. Der private Schlüssel ist in der Datei, die unter dem Attribut DefaultKey abgelegt ist.

Über weitere Informationen über SSL sei ebenfalls auf Sekundärliteratur verwiesen (http://www.openssl.org/).

Das SSL-Modul kann Client Zertifikate auswerten und verifizieren. Das Attribut

TrustedCAPath gibt an, wo im System die CA Zertifikate liegen.

Nähere Informationen finden sich in der gesonderten SSL-Dokumentation.

5.3 MWSL Funktionen

5.3.1 AddHTTPHeader

Syntax	`AddHTTPHeader(<Http-Header>)` Mit diesem Befehl können aus MWSL heraus HTTP-Header hinzugefügt werden. Diese werden dann nicht als Teil des Dokuments, sondern im Protokoll Teil von HTTP übertragen.	
Parameter	`<Http-Header>`	Zeichenkette, die mit \r\n endet. Sollen mehrere HTTP-Header eingetragen werden, so sind die einzelnen Header durch \r\n zu trennen.

Beispiel MWSL	`<MWSL><!--`
	`  var strCookie;`
	`  strCookie = "Set-cookie: siemens_automation_language=";`
	`  strCookie = strCookie + GetVar( "Language", "URL");`
	`  strCookie = strCookie + ", path=/\r\n";`
	`  AddHTTPHeader( strCookie );`
	`--></MWSL>`	

5.3.2 CacheVar

<table>
<tr><td>Syntax</td><td colspan="2">CacheVar(<Variablennamen>)

Diese Funktion legt Prozessvariablen in einem Cache-Speicher ab.
Bei nochmaligem Zugriff auf die Variablen müssen diese nicht erneut von der Variablenquelle geholt werden. Außerdem kann damit ein Prozessabbild zu einem bestimmten Zeitpunkt generiert werden.
Bei einem erneuten Aufruf von CacheVar() wird der Wert der Variablen aktualisiert.
Der Cache-Speicher ist nur für die jeweilige MWSL-Seite gültig.</td></tr>
<tr><td>Parameter</td><td><Variablenname></td><td>String mit den Namen der Variable.
Trennzeichen ist das Pipe-Zeichen (" | ")</td></tr>
<tr><td>Beispiel</td><td colspan="2">

```
<MWSL><!--
  CacheVar("MiniWeb_Build|WWWRoot");
  //  Die Prozessvariablen MiniWeb_Build und WWWRoot werden
  //  zum Cache hinzugefügt.

  WriteVar("MiniWeb_Build");
  WriteVar("WWWRoot");

  CacheVar("SystemTime");
  //  SystemTime wird zum Cache hinzugefügt.

  WriteVar("SystemTime");

  CacheVar("SystemTime");
  //  Der Wert von SystemTime wird erneuert.

  WriteVar("SystemTime");

--></MWSL>
```

Wenn während der Verarbeitung der MWSL-Seite ein Wert geschrieben wird, dann wird er im Cache und im Prozess aktualisiert.
</td></tr>
</table>

5.3.3 ExistVariable

<table>
<tr><td>Syntax</td><td colspan="2"><code>ExistVariable(<Variablenname>, <Variablenquelle>)</code>

Dieser Befehl fragt das Vorhandensein einer Variablen ab.
Er liefert TRUE oder FALSE zurück.</td></tr>
<tr><td rowspan="2">Parameter</td><td><code><Variablenname></code></td><td>Name der Variable</td></tr>
<tr><td><code><Variablenquelle></code></td><td>Name der Variablenquelle
Mögliche Quellen: "URL", "PROCESS", "HTTP", "COOKIE"
Wird dieser Parameter weggelassen, so nimmt <code>ExistVariable</code> als Default den Quelle "PROCESS"</td></tr>
<tr><td>Beispiel</td><td colspan="2"><code>ExistVariable("Parameter", "URL")</code> Existiert die URL Variable "Parameter", so wird TRUE zurückgegeben, andernfalls FALSE.

<code>ExistVariable("Color", "PROCESS")</code> Existiert die Prozessvariable "Color", so wird TRUE zurückgegeben, andernfalls FALSE.

<code>ExistVariable("Color")</code> Existiert die Prozessvariable "Color", so wird TRUE zurückgegeben, andernfalls FALSE.</td></tr>
<tr><td>Beispiel</td><td colspan="2">Meistens wird ExistVariable in einer if-Bedingung abgefragt, um entsprechend zu reagieren, falls eine Variable nicht vorhanden ist (siehe If (Seite 178)).
<pre><MWSL><!--
  var fakultaet = 1;
  var fakul = 0;
  var zaehler = 0;
  if(ExistVariable("Fakul", "URL"))
  {
    fakul = GetVar("Fakul", "URL");
  }
  for(zaehler = 1; zaehler <= fakul; zaehler++)
  {
     fakultaet = fakultaet * zaehler;
  }
  write ("F(" + fakul + ") = " + fakultaet);
--></MWSL></pre>
Wenn die URL-Variable Fakul existiert, wird die lokale Variable fakul mit dem Wert der URL-Variablen belegt.

Achtung!
Die Variablenquelle URL liefert für nicht vorhandene URL-Parameter immer eine Leerkette, daher können Variablen von der Variablenquelle URL nicht mit ExistVariable() abgefragt werden, ExistVariable() liefert dann immer TRUE.
Es ist aber möglich, auf die Leerkette abzufragen:

<pre>If (GetVar("MyVar", "URL") == "")
{
   write("Url Parameter MyVar not set.");
}</pre></td></tr>
</table>

5.3.4 GetVar

Syntax	`GetVar(<Variablenname>, <Variablenquelle>, <Formatstring>);` Diese Funktion liefert den Wert einer Variablen von einer Variablenquelle zurück. Existiert ein Parameter nicht, so wird "null" zurückgeliefert.	
Parameter	`<Variablenname>`	Name der Variable
	`<Variablenquelle>`	Name der Variablenquelle Die genannten Variablenquellen sind die vom Webserver mitgelieferten. Es können weitere Variablenquellen entwickelt werden, die analog zu behandeln sind. Wird dieser Parameter weggelassen, so nimmt `GetVar()` als Default den Typ `PROCESS`.
	`<Formatstring>`	Die Handhabung des Formatstring ist abhängig von der Variablenquelle. So ist diese Eigenschaft für die Variablenquellen `COOKIE` und `URL` nicht möglich. Syntax einer HTTP-Variablen: Variablen und HTTP Header Angaben (Seite 174) Syntax einer Prozessvariablen: Globale Variablen (Seite 168)
Beispiel	`GetVar("Color");` Gibt den Inhalt der Variablen `Color` zurück. Da `PROCESS` die Defaultvariablenquelle ist, entspricht das Ergebnis dem des folgenden Aufrufs. `GetVar("Color", "PROCESS");` Gibt den Inhalt der Variablen Farbe, der Variablenquelle `PROCESS` zurück. `GetVar("Parameter", "URL");` Gibt den Inhalt der Variablen `Parameter` aus der URL zurück. `GetVar("Accept-Language", "HTTP", "?-" )` Gibt den Inhalt der HTTP-Variablen `Accept-Language` zurück. Der Formatstring "?-" gibt an, dass alle Zeichen bis zum ersten Auftreten des "-" Zeichens zurückgegeben werden. `GetVar("Color", "PROCESS", "[2,3]");` Gibt 3 Zeichen ab der Position 2 der Prozessvariablen `Color` zurück. Das Ergebnis sind die Zeichen 2-5 der Prozessvariablen. `GetVar("Accept-Language", "HTTP", "[3,0]" )` Gibt den Inhalt der HTTP-Variablen `Accept-Language` ab dem 3 Zeichen zurück.	

5.3.5 InsertFile

Syntax	`InsertFile(<Dateiname>)` Mit diesem Befehl lässt sich eine vorhandene Textdatei einfach importieren. Diese Textdatei wird vor dem Einfügen mit MWSL interpretiert.	
Parameter	`<Dateiname>`	Name der Textdatei inklusive Pfad.
Beispiel	`<HTML>` `  <HEAD>` `  </HEAD>` `  <BODY>` `  [...]` `  <table>` `  [...]` `  <tr>` `  <td>` `  Auf der rechten Seite wird nun eine HTML-Datei angezeigt.` `  </td>` `  <td>` `   ` `  </td>` `  <td>` `  <MWSL>InsertFile("/MWSL/Output.mcs")</MWSL>` `  </td>` `  </tr>` `  [...]` `  </table>` `  [...]` `  </BODY>` `</HTML>` In der rechten Spalte der Tabelle wird nun die HTML-Datei `Output.mcs` eingefügt und angezeigt.	

5.3.6 ProcessXMLData

Syntax	`ProcessXMLData( <DATA>, <TEMPLATE>)` Mit diesem Befehl kann man anhand von Daten- und Templatedateien dynamische HTML-Dateien generieren. Man hat eine Datei, in der die Daten vorhanden sind (Data), und eine andere Datei, in der die Struktur definiert ist (Template). ProcessXMLData() fügt die beiden Dateien zu einer HTML-Datei zusammen. Dabei wird für jedes Datum die Template-Datei durchgegangen, um festzustellen, wie es angezeigt werden muss. Man erreicht dadurch eine Trennung von Daten und Inhalt. Mit einer nachträglichen Änderung der Template Datei kann man das Aussehen einer oder vieler Seiten ändern, ohne die Datenseiten ändern zu müssen. Somit lassen sich leichter Daten ergänzen, und es lässt sich auch wesentlich leichter eine andere Strukturierung aufbinden, da man einfach eine andere Template-Datei einfügen muss. Mehr zum Template- Mechanismus: Funktionsweise des Template Mechanismus (Seite 181)	
Parameter	`<DATA>`	Daten für die dynamische HTML-Datei Als Parameter kann eine Datei oder eine Variable, in der sich die Daten befinden, übergeben werden. Datei: "`<EXTERNAL SRC=\"/datafile.xml \"/>`", wobei `datafile.xml` die Datei ist, in der sich die Daten befinden. Variable: `<Variablenname>` Es wird einfach der Variablenname angegeben.
	`<TEMPLATE>`	Template (Wie werden die Daten angezeigt) Als Parameter kann eine Datei oder eine Variable, in der sich die Templates befinden, übergeben werden. Datei: "`<TEMPLATES><EXTERNAL SRC=\"/Template.xml\"/> </TEMPLATES>`", wobei es sich bei "Template.xml" um die Datei handelt, in der sich die Templates befinden. Variable: `<Variablenname>` Es wird einfach der Variablenname angegeben.

Beispiel	`ProcessXMLData("<EXTERNAL SRC=\"/MWSL/variables.xml \"/>", "<TEMPLATES><EXTERNAL SRC=\"/MWSL/variablesTemplate.xml\"/></TEMPLATES>");`

Beispiel MWSL	`<MWSL><!--`
	`var Head = "<Provider Name =\"MyVarProvider\">";`
	`var Data = "<Variable Name=\"ZUFUEHRUNG\" Type=\"String\" InitialValue=\"good\" Behavior=\"Manual\" Description=\"Teilezufuehrung.\"/>";`
	`var Foot = "</Provider>";`
	
	`var XMLData = Head + Data + Foot;`
	
	`var TemplateHead = "<TEMPLATES>";`
	`var TemplateFoot = "</TEMPLATES>";`
	`var TemplateFile = "<EXTERNAL SRC=\"" + GetVar("Template", "URL") + "\"/>";`
	`ProcessXMLData(XMLData, TemplateHead + TemplateFile + TemplateFoot);`
	`--></MWSL>`
	
	In der Variablen XMLData sind die Datenknoten mit den zugehörigen Attributen enthalten.
	Zu beachten ist, dass man die " mit \ vor dem XML Parser schützt.
	Auf die gleiche Art und Weise kann man auch eine Variable für die Templates definieren.
	Im oben gezeigten Beispiel besteht die Variable für die Templates aus einer Dateiangabe.

5.3.7 SetVar

Syntax	`SetVar(<Variablenname>, <Wert>)` Mit dieser Funktion werden Prozessvariablen gesetzt. Welche Variablen zu schreiben sind, hängt von den Variablenquellen ab. Aus Sicherheitsgründen können nur Variablen gesetzt werden, die mit dem Prefix "USER_" oder "SESSION_" beginnen, z. B. "SESSION_Parameter".	
Parameter	`<Variablenname>`	Name der Variable Es ist zu beachten, dass der Variablenname zwingend mit `USER_` oder `SESSION_` beginnen muss. z. B. "`SESSION_Parameter`"
	`<Wert>`	Der neue Variablenwert.
Beispiel	`SetVar("Color", "Green");` Setzt die Prozess Variable `Color` auf den Wert `Green`.	

5.3.8 ShareRealm

Syntax	`ShareRealm(<Gruppe>)` Gibt an, ob der aktuelle User ein Mitglied der als Parameter übergebenen Gruppe ist. Der Rückgabewert kann `TRUE` oder `FALSE` sein.	
Parameter	`<Gruppe>`	Als Parameter sind momentan folgende Ausprägungen zulässig: • `NO_REALM` Keine Gruppenzugehörigkeit • `ANY_REALM` Irgendeine Gruppenzugehörigkeit • `[Gruppenname]` Mitglied der Gruppe `[Gruppenname]` Gruppen sind konfigurationsabhängig.
Beispiel	`write(ShareRealm ("ANY_REALM"));` Ist der aktuelle Benutzer in irgendeiner definierten Gruppe, so wird 1 ausgegeben. Andernfalls 0.	

Beispiel MWSL	`<MWSL><!--`
	` if ( ShareRealm("ANY_REALM"))`
	`  {`
	`  write ("<tr valign=\"baseline\">\r\n");`
	`  write ("<td><H2>Hello " + GetVar( "Username",  "HTTP") +  ", you're successfully logged in.</H2></td>\r\n");`
	`  write ("</tr>\r\n");`
	`  }`
	`--> </MWSL>`
	Falls der Benutzer Mitglied in irgendeiner Gruppe ist, werden die Anweisungen in den geschweiften Klammern ausgeführt.	

5.3.9 write

Syntax	`write(<Text>)` Die Funktion write() schreibt Text in die Ausgabe einer HTML-Seite.	
Parameter	`<Text>`	Es können Text, Rückgabewerte von Funktionen oder Variableninhalte übergeben werden.

Beispiel	`<MWSL><!--`
	`write("Hello World");`
	`// Ausgabe: Hello World`
	
	`write("Hello" + " " + "World");`
	`// Ausgabe: Hello World`
	
	`write(GetVar("Parameter", "URL"));`
	`// Ausgabe des Inhalts der Variablen Parameter in der URL.`
	
	`write(5+6);`
	`// Ausgabe: 11`
	
	`var zahl1 = 5;`
	`var zahl2 = 7;`
	`var string = "Hello";`
	`write(string + ": " + zahl1 + zahl2);`
	`// Ausgabe: Hello: 57`
	
	`write(zahl1 + zahl2);`
	`// Ausgabe: 12`
	
	`write("Inhalt von Parameter: " + GetVar("Parameter", "URL"));`
	`// Ausgabe: Inhalt von Parameter: Hello`
	`// falls Parameter den String "Hello" beinhaltet.`
	`--></MWSL>`	

5.3.10 WriteVar

Syntax	`WriteVar(<Variablenname>, <Variablenquelle>, <Formatstring>);`
	
	Dieser Befehl gibt den Inhalt einer Variablen aus, er wird in die Ausgabe geschrieben.
	`WriteVar` ist fast die gleiche Funktion wie `GetVar`.
	`WriteVar` ist äquivalent zu dem Aufruf:
	
	`WriteVar(…) === write( GetVar ( …) )`
	
	Der einzige Unterschied ist, dass `WriteVar` den Inhalt der angegebenen Variablen ausgibt, während `GetVar()` den Inhalt als Rückgabewert zurückliefert.	

Parameter	`<Variablenname>`	Name der Variable
	`<Variablenquelle>`	Name der Variablenquelle Die genannten Variablenquellen sind die vom Webserver mitgelieferten. Es können weitere Variablenquellen entwickelt werden, die analog zu behandeln sind. Wird dieser Parameter weggelassen, so nimmt `GetVar()` als Default den Typ `PROCESS`.
	`<Formatstring>`	Die Handhabung des Formatstring ist abhängig von der Variablenquelle. So ist diese Eigenschaft für die Variablenquellen `COOKIE` und `URL` nicht möglich. Die Aufrufsyntax ist äquivalent zu der von GetVar() (Seite 297) .

Beispiel

```
<MWSL><!--   write(GetVar("Parameter", "URL"));
   // Ausgegeben wird hier der Inhalt der Variablen Parameter.
   // GetVar liefert den Wert der Variablen,
   // der dann mit write in die Ausgabe geschrieben wird.
   // (Siehe auch GetVar() und write()).
   // Die Gleiche Ausgabe kann auch mit folgendem Befehl
   // realisiert werden.
   WriteVar("Parameter", "URL");
   // WriteVar schreibt direkt in die Ausgabe und liefert
   // keinen Returnwert


   WriteVar("Color");
   // Gibt den Inhalt von Color, welches eine
   // Prozessvariable ist, aus.


   WriteVar("Accept-Language", "HTTP", "?-" )
   // Gibt den Inhalt der HTTP-Variablen "Accept-Language"
   // bis zum "-" Zeichen aus.


   WriteVar("Color", "PROCESS", "[2,3]");
   // Gibt die Zeichen 2-5 der Prozessvariablen Color aus.


   WriteVar("Accept-Language", "HTTP", "[3,0]" )
   // Gibt den Inhalt der HTTP-Variablen "Accept-Language"
   // ab dem 3 Zeichen aus.
--> </MWSL>
```

Siehe auch

Globale Variablen (Seite 168)

5.3.11 WriteXMLData

WriteXMLData()

Syntax	`WriteXMLData( <DATA>, <TEMPLATE>)` WriteXMLData gibt die Daten im Gegensatz zu ProcessXMLData aus. Statt `write( ProcessXMLData(…));` kann man auch `WriteXMLData(…);` schreiben. `WriteXMLData();` bekommt die gleichen Parameter wie `ProcessXMLData();`.	
Parameter	`<DATA>`	Daten für die dynamische HTML-Datei Als Parameter kann eine Datei oder eine Variable, in der sich die Daten befinden, übergeben werden. Datei: "`<EXTERNAL SRC=\"/Datendatei.xml \"/>`", wobei Datendatei die Datei ist, in der sich die Daten befinden. Variable: `<Variablenname>` Es wird einfach der Variablenname angegeben.
	`<TEMPLATE>`	Template (Wie werden die Daten angezeigt) Als Parameter kann eine Datei oder eine Variable, in der sich die Templates befinden, übergeben werden. Datei: "`<TEMPLATES><EXTERNAL SRC=\"/Template.xml\"/></TEMPLATES>`", wobei es sich bei "Template.xml" um die Datei handelt, in der sich die Templates befinden. Variable: `<Variablenname>`

Beispiel	`WriteXMLData("<EXTERNAL SRC=\"/MWSL/variables.xml \"/>",`
	`"<TEMPLATES><EXTERNAL`
	`SRC=\"/MWSL/variablesTemplate.xml\"/></TEMPLATES>");`
	
	Mehr zum Template Mechanismus: Funktionsweise des Template Mechanismus (Seite 181)	

5.3.12 NodeIndex

Variable	`NodeIndex`	`NodeIndex` ist eine Prozess-Variable, die beim Parsen eines Templates zur Verfügung steht. Diese Variable gibt die Anzahl der bereits durchlaufenen Knoten aus. Der Zugriff erfolgt genauso wie bei anderen Variablen der `PROCESS`-Variablenquelle. Mehr zum Template Mechanismus: Funktionsweise des Template Mechanismus (Seite 181)

Beispiel	`<?xml version="1.0" ?>`
	`<TEMPLATES>`
	`<TEMPLATE NAME="Variable">`
	`<POSITION NAME="LINE">`
	`<![CDATA[`
	`<MWSL> WriteVar("NodeIndex ") </MWSL>`
	`]]>`
	`</POSITION>`
	`</TEMPLATE>`
	`</TEMPLATES>`	

5.3.13 NodeLevel

Variable	`NodeLevel`	`NodeLevel` ist eine Prozess-Variable, die beim Parsen eines Templates zur Verfügung steht. Diese Variable gibt die Hierarchieebene des aktuellen Knotens aus. Der Zugriff erfolgt genauso wie bei anderen Variablen der `PROCESS`-Variablenquelle. Mehr zum Template Mechanismus: Funktionsweise des Template Mechanismus (Seite 181)

Beispiel	`<?xml version="1.0" ?>`
	`<TEMPLATES>`
	`<TEMPLATE NAME="Variable">`
	`<POSITION NAME="LINE">`
	`<![CDATA[`
	`<MWSL> WriteVar("NodeLevel") </MWSL>`
	`]]>`
	`</POSITION>`
	`</TEMPLATE>`
	`</TEMPLATES>`	

5.4 IT DIAG Dateien

5.4.1 DIAGURLS.TXT

Aufbau der Datei DIAGURLS.TXT

Die Datei, die sich im Verzeichnis /HMI/SYSLOG/DIAG befindet, enthält die Namen der IT DIAG-Seiten, die gesichert werden sollen.

Hier ein Beispiel für das Aussehen dieser Datei:

```
alarms.mcs
alarmsdrv.mcs
alarmbuf.mcs
devinfo.mcs
diagbuff.mcs
diagbuffdrv.mcs
diagnost.mcs
ipconfig.mcs
mempool.mcs
start.mcs
taskrunt.mcs
timezone.mcs
```

Inhalt der Datei DIAGURLS.TXT

Siehe auch

Diagnostic files (Seite 51)

5.5 Ländercodes LCID

5.5.1 Tabelle LCID

Länderspezifische Codes

Dezimalwert	Land	UMC-Kürzel	Priorität
1033	English - United States	B	1
2057	English - Great Britain	B	2
1031	German - Germany	A	3
1036	French - France	C	4
1034	Spanish - Spain (Trad.)	D	5
1040	Italian - Italy	E	6
3081	English - Australia	B	10
10249	English - Belize	B	10
4105	English - Canada	B	10
9225	English - Caribbean	B	10
6153	English - Ireland	B	10
8201	English - Jamaica	B	10
5129	English - New Zealand	B	10
13321	English - Phillippines	B	10
7177	English - Southern Africa	B	10
11273	English - Trinidad	B	10
3079	German - Austria	A	20
5127	German - Liechtenstein	A	20
4103	German - Luxembourg	A	20
2055	German - Switzerland	A	20
2060	French - Belgium	C	30
3084	French - Canada	C	30
5132	French - Luxembourg	C	30
4108	French - Switzerland	C	30
11274	Spanish - Argentina	D	40
16394	Spanish - Bolivia	D	40

13322 Spanish - Chile D 40

9226 Spanish - Colombia D 40

5130 Spanish - Costa Rica D 40

7178 Spanish - Dominican Rep.D 40

12298 Spanish - Ecuador D 40

17418 Spanish - El Salvador D 40

4106 Spanish - Guatemala D 40

18442 Spanish - Honduras D 40

2058 Spanish - Mexico D 40

19466 Spanish - Nicaragua D 40

6154 Spanish - Panama D 40

15370 Spanish - Paraguay D 40

10250 Spanish - Peru D 40

20490 Spanish - Puerto Rico D 40

14346 Spanish - Uruguay D 40

8202 Spanish - Venezuela D 40

2064 Italian - Switzerland E 50

1078 Afrikaans

1052 Albanian

14337 Arabic - United Arab Emirates

15361 Arabic - Bahrain

5121 Arabic - Algeria

3073 Arabic - Egypt

2049 Arabic - Iraq

11265 Arabic - Jordan

13313 Arabic - Kuwait

12289 Arabic - Lebanon

4097 Arabic - Libya

6145 Arabic - Morocco

8193 Arabic - Oman

16385 Arabic - Qatar

1025 Arabic - Saudi Arabia

10241 Arabic - Syria

7169 Arabic - Tunisia

9217 Arabic - Yemen

1067 Armenian

1068 Azeri - Latin

2092 Azeri - Cyrillic

1069 Basque

1059 Belarusian

1026 Bulgarian

1027 Catalan

2052 Chinese - China

3076 Chinese - Hong Kong SAR

5124 Chinese - Macau SAR

4100 Chinese - Singapore

1028 Chinese - Taiwan

1050 Croatian

1029 Czech

1030 Danish

1043 Dutch - Netherlands

2067 Dutch - Belgium

1061 Estonian

1065 Farsi

1035 Finnish

1080 Faroese

2108 Gaelic - Ireland

1084 Gaelic - Scotland

1032 Greek

1037 Hebrew

1081 Hindi

1038 Hungarian

1039 Icelandic

1057 Indonesian

1041 Japanese

1042 Korean

1062 Latvian

1063 Lithuanian

1071 F.Y.R.O. Macedonia

1086 Malay - Malaysia

2110 Malay – Brunei

1082 Maltese

1102 Marathi

1044 Norwegian - Bokml

2068 Norwegian - Nynorsk

1045 Polish

2070 Portuguese - Portugal

1046 Portuguese - Brazil

1047 Raeto-Romance

1048 Romanian - Romania

2072 Romanian - Republic of Moldova

1049 Russian

2073 Russian - Republic of Moldova

1103 Sanskrit

3098 Serbian - Cyrillic

2074 Serbian - Latin

1074 Setsuana

1060 Slovenian

1051 Slovak

1070 Sorbian

1072 Southern Sotho

1089 Swahili

1053 Swedish - Sweden

2077 Swedish - Finland

1097 Tamil

1092 Tatar

1054 Thai

1055 Turkish

1073 Tsonga

1058 Ukrainian

1056 Urdu

2115 Uzbek - Cyrillic

1091 Uzbek – Latin

1066 Vietnamese

1076 Xhosa

1085 Yiddish

1077 Zulu

Index

V

W

Z